제2권 통권 6호 1989년 5월 1일 발행(매월 1일 발행) 1988년 10월 10일 등록(등록번호 라-3794) (우)140-150 서울특별시 용산구 갈월동 17

퍼스널컴퓨터

하이테크

89/05

특집/ MSDOS.SYS의 분석

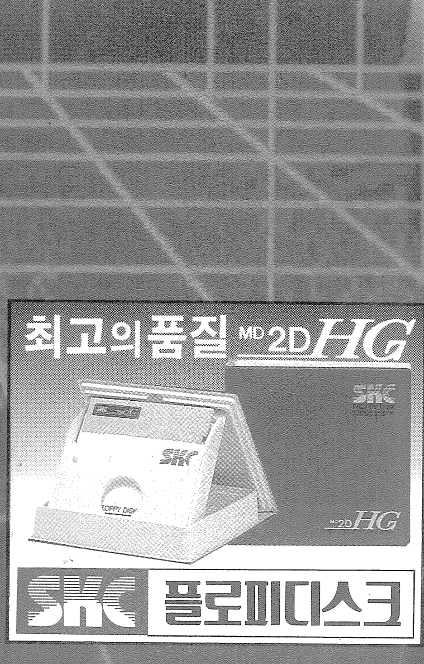
MS-DOS의 구성/내부 인터럽트기능/
프로그램의 실행과 종료/프로세스의 기동과 종료

활용/ 디렉토리의 결합과 분리 /하드디스크에서의 파킹/파스칼에서의 지수계산/
삼보컴퓨터의 파킹 프로그램 TG SHIP.COM의 분석

테크닉/ 마우스 드라이버 툴키트/마우스를 이용한 그래픽에디터 SEM-DRAW/
쿼트로의 폰트를 터보 파스칼에서 이용한다.



공개 소프트웨어소개/
디렉토리에 관한 모든 기능을 가진
별난 소프트웨어 DX.COM
워드프로세서/ 발송문 관리



□편집자노트□

올바른 컴퓨터교육이 이루어질 수 있어야 합니다.

거리 풍경이 화사하게 밝아지는가 싶더니 며칠째 30도를 오르내리는 무더운 날씨가 계속됩니다.겨울에서 봄을 거치지 않고 바로 여름으로 직행하지 않나 싶을 정도로 짜증스럽게 여겨지는 날씨입니다.

기상대발표로는 80몇년만에 처음있는 이상고온이며 올여름에는 무더위가 일찍오고 또 유난히 길 것이라고 합니다.

독자여러분도 건강에 주의하시기 바랍니다. 얼마전에 발표된 2000년대 정보화사회를 대비하여 초·중등학교 학생 전체에게 컴퓨터 교육을 실시할 것이라는 정부의 계획이 시행단계에서 자꾸만 여러가지 문제점들이 들어나 지연되고 있습니다.

그런데 그 지연사유가 각급학교 컴퓨터교육 계획의 입안시부터 문제점으로 지적된 기종선정에 관한 문제 때문이라고 합니다.

당초 계획입안시에 지난번의 시행착오를 거울삼아 기종간의 호환성 문제와 표준화에 대한 의견이 강하게 대두되었으나 결정단계에서는 어찌된 일인지 흐지부지 되어버리고 결국 초·중등학교에서는 8비트 기종을 보급하기를 확정하고 이미 일부 학교에는 보급까지 완료된 상태입니다. 얼마전에도 편집자노트를 통해 호환성이 결여된 기종을 선택하므로써 발생되는 문제점과 대책방안에 대해서 거론한적이 있습니다.

그런데 그러한 문제가 실제로 표면화되어 시행단계에서부터 일선 교육관계자와 일부 업계 관계자들 사이에서는 강하게 반론을 제기하고 있습니다.

이들은 호환성이 없는 8비트 기종을 보급할 경우 각급 학교간의 컴퓨터교육에 혼선을 빚을 뿐 아니라 다가오는 정보화사회에 대비하여 제대로 연계교육이 불가능하다는 점을 지적하고 있습니다.

다행히 관련 정부당국에서 이러한 문제점들을 검토하여 교육용기종으로 호환성 문제와 표준화 작업을 고려하여 16비트기종을 선택하는 문제를 신중히 검토하고 있다고 합니다.

그런데 막상 이러한 움직임이 보이자 8비트 기종 보급업체에서는 선진국에서도 교육용으로 8비트 기종을 사용하고 있는 점과 16비트 기종으로 대체될 경우 기존에 보급된 8비트 기종은 사장될 것이라는 점들을 들어 강하게 반발하고 있다고 합니다. 누구말이 옳고 그르고를 떠나서 이러한 여러가지의 문제점들이 발생된 원인은 정책입안시 이러한 점을 충분히 고려하여 신중히 선택하여야 함에도 불구하고 그러지 못하였기 때문이라고 볼 수 있습니다.결국 이렇게 되면 피해를 입는 사람은 장래 2000년대 정보화사회를 이끌어 나갈 지금의 학생들이라고 볼 수 있습니다. 정책이 잘못될 경우 이들이 입는 피해는 누가 책임져야 되는지 모두가 이점을 고려하여 정책을 결정하였으면 합니다.

그렇게 하여야만 올바른 컴퓨터교육이 이루어질 수 있을 것입니다.

어떻게 시작하나요 6
파이핑과 필터

파이핑과 필터

이번달에는 지난달의 표준 입출력의 재지정에 이어서 한 프로그램의 출력이 다른 프로그램의 입력이 될 때 관계하는 파이프(Pipe)와 필터(Filter)의 기능에 관하여 공부해보자.

파이핑

파이핑은 한 프로그램의 표준출력을 다른 프로그램의 표준입력으로 사용하도록 하는 기능이다. 이를 위해 DOS는 일시적으로 화일을 만들어 파이프된 입출력 데이터를 보관하며, 일시적인 화일은 디폴트 드라이브의 루트디렉토리에 작성된다.

다음과 같이

```
DIR > %PIPE.$$$
MORE < %PIPE.$$$
```

라고 하면 한 화면씩 디렉토리의 내용이 다음과 같이 디스플레이된다.

```
Volume in drive C has no label
Directory of  C:\DOS\DOS3_3

.            <DIR>      8-12-88   3:26p
..           <DIR>      8-12-88   3:26p
4201     CPI     17089  3-18-87  12:00p
5202     CPI       459  3-17-87  12:00p
A                 8048  1-04-80  11:33p
ANSI     SYS      1678  3-17-87  12:00p
APPEND   EXE      5825  3-17-87  12:00p
ASSIGN   COM      1561  3-17-87  12:00p
ATTRIB   EXE      9529  3-17-87  12:00p
BACKUP   COM     31913  3-18-87  12:00p
BASIC    COM       941  1-01-80  12:46a
BASIC    PIF       369  3-17-87  12:00p
BASICA   EXE     54272  5-13-83  12:00p
KANG     BAS        65  1-04-80   7:40p
CHKDSK   COM      9850  3-18-87  12:00p
CLOUD    COM      8048  1-04-80  11:33p
COMMAND  COM     25307  3-17-87  12:00p
COMP     COM      4214  3-17-87  12:00p
COUNTRY  SYS     11285  3-17-87  12:00p
DEBUG    COM     15897  3-17-87  12:00p
-- More --
```

이 명령은 한 화일에 재지정된 첫번째 프로그램의 출력이 두번째 프로그램의 재지정된 입력으로 사용되는 것과 같다고 생각할 수 있다.

이와 같이 파이핑은 지난달에 공부했던 표준 입출력의 재지정 기능을 마치 사슬로 엮는 것과 같은 연쇄기능(Chain)을 갖는다. 이 명령은 다음과 같이

```
DIR ¦ MORE
```

라고 한 것과 같음을 실행을 통하여 알 수 있다. 여기서 한가지 주의할 점은 디스크에 화일을 만들 공간이 없을 때는 도스가 일시적인 파이핑화일을 만들지 못하므로 에러가 발생한다. 그리고 파이핑되는 프로그램을 지우거나 수정해서는 안된다.

필터

필터는 표준 입력장치로부터 데이터를 읽어들여 수정한 후 처리된 결과를 표준출력장치로 다시 내보내는 프로그램이나 명령어이다.

필터라는 뜻이 '여과된다'는 의미라고 볼 때 이 기능은 데이터가 프로그램에 의해 여과된다고 생각하면 이해가 빠를 것이다. 필터에는 다음과 같은 3가지의 표준필터가 있다.

- FIND : 특정한 문자열을 한 화일에서 찾아 이를 표준출력장치(모니터)에 내보낸다.
- MORE : 한 화면에 데이터를 내보내고, 유저가 더 디스플레이하라는 키를 누를 때까지 기다린다.
- SORT : 아스키 순서에 따라 화일을 소트한다.

(1) FIND 명령

이 명령은 지정된 화일에서 지정된 문자열을 갖는 모든 라인을 찾아서 표준 출력장치로 보내는 기능을 수행한다.

디렉토리내에 있는 특정한 문자열을 갖는 화일을 찾고자 한다면 DIR명령과 함께 수직바(|)를 한 다음, FIND의 오른쪽에 찾고자 하는 문자열을 준다.

다음과 같이

```
DIR ¦ FIND "3-05-89"
```

하게 되면 DIR 명령에 의해 현재의 디렉토리목록이 만들어지며, 디렉토리목록은 화면에 나타나는 대신에 FIND 명령의 입력화일로써 보내진다. 그리고 FIND 필터는 "3-05-89"에 만들어진 화일만을 다음과 같이 디스플레이시킨다.

```
C:\DOS\DOS3_3>DIR ¦ FIND "3-05-89"
```

```
LIST     BAS    1103  3-05-89  11:49a
LIST2    BAS    1082  3-05-89  11:18a
LIST1    BAS    1074  3-05-89  11:34a
AZ       BAS     305  3-05-89   4:54p
Z        BAS       0  3-05-89   3:47p
```

다음 예는 FIND 필터가 화일명이나 확장자에 특정 문자열을 갖는 화일을 나타내게 하는 것으로 다음과 같이

```
DIR ¦ FIND "EXE"
```

하게 되면, EXE라는 문자열을 확장자로 갖는 화일을 디스플레이 한다.

```
C:\DOS\DOS3_3>DIR ¦ FIND "EXE"
APPEND   EXE     5825  3-17-87  12:00p
ATTRIB   EXE     9529  3-17-87  12:00p
BASICA   EXE    54272  5-13-83  12:00p
FASTOPEN EXE     3919  3-17-87  12:00p
FIND     EXE     6434  3-17-87  12:00p
H        EXE    29184  1-01-80  12:06a
JOIN     EXE     8969  3-17-87  12:00p
NLSFUNC  EXE     3060  3-17-87  12:00p
REPLACE  EXE    11775  3-17-87  12:00p
SD       EXE     1280 11-11-83  10:10a
SHARE    EXE     8608  3-17-87  12:00p
SORT     EXE     1977  3-17-87  12:00p
SUBST    EXE     9909  3-17-87  12:00p
XCOPY    EXE    11247  3-17-87  12:00p
GWBASIC  EXE    80592  7-24-87  12:00a
BASIC    EXE    66928  4-10-84  10:10a
```

만일 FIND 명령을 입력할 때 파라미터가 잘못 지정되면 "File not find"란 메시지가 나타난다.

FIND 필터는 3가지 옵션 파라미터 /V, /C, /N 등을 사용할 수 있는데 FIND 명령 바로 뒤에 위치하며, 찾고자 하는 문자열 앞에 넣어야 한다.

즉, 다음과 같이 입력을 해야 한다.

```
FIND /V "BAT"
```

만일, 여기서 문자열은 주지않고 /V, /C, /N만을 입력하면 DOS는 "V", "C", 또는 "N"으로 시작하는 화일을 찾게 된다. 그리고 찾으려는 문자열(S-tring) 은 반드시 이중 따옴표("")로 묶어야 한다. 만일 단일 따옴표(' ')를 2개 연속적으로 찍으면 단일 따옴표(")로 간주되므로 유의하여야 한다.

각 파라미터의 기능은 다음과 같다.

/V : 지정한 문자열을 갖지 않는 모든 라인을 표시한다.

/C : 지정한 문자열이 포함되어 있는 모든 라인의 총 수만을 표시한다.

/N : 지정된 문자열을 찾아 그 문자열이 있는 라인의 번호와 그 라인을 디스플레이시킨다. 만약 /C와 /N을 동시에 지정하면 FIND는 /N을 무시하고 /C만 실행한다.

□주의

FIND 필터는 대문자와 소문자를 다르게 취급하므로 문자열이 어떻게 되어 있는지 잘 알아야 한다.

보통 컴퓨터에 문자를 입력할 때 대문자와 소문자를 구별없이 취급하는데 FIND 필터는 대문자와 소문자를 다르게 취급한다. 그러므로 문자열이 소문자로 되어 있는지, 대문자로 되어 있는지를 잘 알아야 한다.

예를 들어, "to"라는 문자열이 나타내는 모든 것을 찾고자 할 때 다음과 같이

```
FIND "To" MEN.TXT
```

라고 하면 대문자와 소문자가 매치되지 못하므로 지정된 화일에서 원하는 문자열을 찾지 못하고 화면에는

```
------------MEN.TXT
```

만 디스플레이된다.

또한, 지시한 문자열이 확실히 구별되지 않으면 불필요한 문자열을 찾아내게 되므로 찾고자 하는 문자열을 독특하게 하여 쉽게 구별할 수 있도록 해야 한다. 예를 들어, 12월을 찾고자 한다면 "12" 대신에 "12-"라고 써야 쉽게 찾을 수 있다. 그렇지 않으면 12와 관계되는 모든 문자열을 찾게 된다.

□주의

/C 파라미터는 한 화일내에서 지정한 문자열의 수만 체크할 뿐 지정한 문자열이 몇개인지는 세지 않는다.

/C 파라미터는 지정한 문자열이 나타나는 라인을 모두 세어 그 라인 수가 얼마인가를 말해준다. 만약 한 라인에 문자열이 3번 나타났다고 하더라도 관계없이 그 라인만 계산한다.

다음 예를 보자. 우선 MEN.TXT라는 화일을 만들어 확인해 보자.

```
COPY CON MEN.TXT
Now is the time for all good men
to come to the aid of their party.
[F6]
```

여기서 "to"가 몇번 나왔는지 알기 위해

```
FIND /C "to" MEN.TXT
```

라고 하면 "to"가 나타나는 라인의 수를 세어 다음과 같이 나타낸다.

```
------------- MEN.TXT : 1
```

FIND 필터는 화일 여러개를 한꺼번에 찾기 위한 명령어로 와일드카드인 "*"과 "?"를 사용할 수 없고, 각각의 화일명과 확장자명을 모두 입력해야 한다.

만약 화일을 지정하지 않으면 FIND 필터명령은 공백상태로 된다. 유저가 FIND하기 위한 화일명을 지정하지 않았다면 FIND 필터는 유저가 키보드를 통해 데이터를 입력하고 있는 것으로 판단한다. 즉 FIND는 데이터를 입력하길 기다리는 동안에 화면은 단지 반짝이는 커서만을 보여준다. 지정한 문자를 입력하면 DOS는 그 문자를 그대로 되돌려준다. 그리고 Ctrl-Z가 나타나면 화일의 끝이라고 판단하여 Ctrl-Z 이후의 내용을 표시하지 않는다.

```
C:\DOS\DOS3_3>FIND "DOOLY"
My name is DOOLY.
My name is DOOLY.
^Z
```

□주의

입·출력 재지정과 FIND 필터를 같은 화일에 쓸 수 없다.

도스는 새로운 화일의 위치를 정하고, 기존의 데이터를 제거하기 때문에 만일 특정한 문자열을 포함한 라인에 존재하는 화일을 변환시키려면 새로운 화일에 앞서 재지정을 해야 한다.

```
FIND "string" OLDFILE > NEWFILE
DEL OLDFILE
REN NEWFILE OLDFILE
```

FIND 필터를 사용하는데 있어 마지막으로 유념할 것은 FIND "string" FIND. EXE를 사용해서는 안된다. 비록, 프로그램 화일과 바이너리 코드화된 화일에서와 같이 FIND 필터는 모든 화일에 대해 실행되지만 FIND 프로그램을 실행시키는 화일인 FIND.EXE 화일에서는 실행되지 않으며 실수로 실행시켰을 경우는 시스템을 다시 부팅시켜야 한다.

(2) MORE

이 필터는 표준입력장치에서 한 화면보다 긴 데이터를 받아들여 화면단위로 나누어 출력시킬 경우 사용하는데 한 화면분의 데이터를 출력시킨 후에는 "--More--"라는 메시지를 나타내고 정지한다.

DOS는 한 라인에 127영문자를 받아들이도록 되어 있는데 모니터의 한 화면은 80문자를 나타내므로 80문자보다 길 경우에는 화면상에 나타나지 않는 부분이 있다. 이럴 경우 MORE 필터를 사용하여 80문자 이상이 되면 다음 라인으로 나타내게 하면 된다. 리스트 1과 같이 CHAR.TXT 화일을 만들었다고 하자.

〔리스트 1〕

```
C:\DOS>COPY CON CHAR.TXT
1234567890AAAAAAAAAA1234567890BBBBBBBBBB1234567890CCCCCCCCCC1234567890DDDDDDDDDD
1234567890EEEEEEEEEE1234567890FFFFFFFFFF1234567
^Z
```

이 화일에 대하여 다음과 같이

```
MORE < CHAR.TXT
```

라고 하면 80문자 다음의 글자가 잘라지지 않고 다음 라인에서 나머지 47자가 나타난다.

MORE 필터는 패스를 지정할 때 "<"와 " "을 사용한다. 한 화면 분량씩 표준 출력장치로 전송할 경우에

```
MORE <TEST.ASM
```

라고 하면 TEST.ASM 화일의 내용이 한번에 한 화면씩 디스플레이 된다. 그리고 다음과 같이

```
TREE ¦ MORE
```

라고 하면 TREE 명령어의 출력이 MORE 필터로 파이프된다. 그리고 디스크 전체의 트리구조가 다음과 같이 나타나고 한 화면이 출력되면 "--MORE--" 라는 메시지가 나타난 후 정지된다.

```
DIRECTORY PATH LISTING

Path: \APROLOG

Sub-directories:  None


Path: \MMY

Sub-directories:  FOR3
                  SOURCE


Path: \MMY\FOR3

Sub-directories:  None


Path: \MMY\SOURCE

-- More --
```

이 상태에서 출력을 계속하려면 아무 키나 누르면 된다.

(3) SORT

이 SORT 필터는 표준 입력장치에서 데이터를 읽어들여 소트한 다음 그 데이터를 표준출력장치에 기록한다. 이때 소트할 수 있는 최대 화일은 63KB이다.

□주의
SORT 필터를 사용할 때는 출력 화일명이 입력 화일명과 같아야 하며, 영문자의 대문자와 소문자는 동일하게 처리한다.

SORT 필터에 사용되는 파라미터는 /R, /+n이 있는데 이들의 기능은 다음과 같다.

/R : 역순으로 소트하는데 쓰인다.

/+n : n번째 컬럼부터 소팅을 시작한다. 이때 n은 정수값이 되고, 만일 n을 지정하지 않으면 1번 컬럼이 디폴트로 지정된다.

다음 예를 보자

```
DIR ¦ SORT /R /+24
```

라고 하면 화일이 만들어진 월, 일, 연도를 포함하는 디렉토리목록을 소트화일에 파이핑하고, 24컬럼에 나타난 달을 역순으로 소트한 후 출력을 다음과 같이 표준출력장치에 보낸다

```
C:\DOS\DOS3_3>DIR ¦ SORT /R/+24
 Directory of  C:\DOS\DOS3_3
 Volume in drive C has no label
       75 File(s)   1089536 bytes free
SD       EXE     1280  11-11-83  10:10a
..           <DIR>      8-12-88   3:26p
.            <DIR>      8-12-88   3:26p
GWBASIC  EXE    80592   7-24-87  12:00a
BASICA   EXE    54272   5-13-83  12:00p
CHAR     TXT      129   4-25-89  12:10p
REVDTDIR DIR     3147   4-25-89  11:59a
MEN      TXT       70   4-25-89  11:48a
%PIPE    $$$     2977   4-25-89  11:36a
REVDTDIR BAT      768   4-12-89  10:38p
WOMEN    TXT       71   4-12-89   9:29p
```

```
BASIC    EXE   66928  4-10-84  10:10a
PC       TXT      20  3-21-89   3:50p
RECOVER  COM    4299  3-18-87  12:00p
FORMAT   COM   11616  3-18-87  12:00p
FDISK    COM   48216  3-18-87  12:00p
EGA      CPI   49065  3-18-87  12:00p
CHKDSK   COM    9850  3-18-87  12:00p
BACKUP   COM   31913  3-18-87  12:00p
4201     CPI   17089  3-18-87  12:00p
XCOPY    EXE
```

재지정, 파이핑, 필터의 활용

지금까지 재지정, 파이핑, 필터에 관하여 공부하였다. 어느정도 그 기본적인 기능이 이해되었으면 3가지 명령을 이용하여 간단한 유틸리티로 이용할 수 있는 배치화일을 작성하여 보자.

화일을 백업할 때 가장 최근에 어떤 화일이 만들어졌는지 알 수 있다면 신속하고 정확하게 할 수 있을 것이다.

리스트2는 표준 입출력의 재지정, 파이핑, 필터(FIND, SORT, MORE)를 조합하여 날짜를 역순으로 나타내게 하여 최근에 작성된 화일을 알 수 있는 배치화일의 리스트이다. 이 리스트를 살펴보면 다음과 같다.

7번행의 내용은 24번 컬럼부터 화일이 생성된 일, 월, 년을 포함하는 디렉토리 리스트이다. 이 명령은 디렉토리 리스트를 생성시키고 SORT 필터로 파이프한다. SORT 필터는 월과 일의 역순으로 디렉토리를 소트한 후 그 출력을 REVDTDIR.WRK이라는 워크화일로 재지정한다.

8번행의 경우에서 살펴보면 새로운 디렉토리는 FIND 명령어에 의해 각 라인별로 검색된다. 맨 처음 라인은 "Volume"라인으로 ">"로 재지정된 새로운 디렉토리 화일 REVDTDIR.DIR을 생성한다.

10번행에서 19번행까지는 월과 일의 역순으로 된 워크화일을 가지고 연도의 역순으로 된 레코드를 찾아 연-월-일이 역순으로 된 화일을 만든다.

〔리스트 2〕

```
1  REM*********************************************
2  REM*                                           *
3  REM*****        REVDTDIR.BAT              ******
4  REM*****     REVERSE  DATE  DIRECTORY     ******
5  REM*                                           *
6  REM*********************************************
7  DIR ¦ SORT /R/+24 > REVDTDIR.WRK
8  FIND "Volume in"   REVDTDIR.WRK > REVDTDIR.DIR
9  FIND "Directory"   REVDTDIR.WRK >> REVDTDIR.DIR
10 FIND "-89"     REVDTDIR.WRK >> REVDTDIR.DIR
11 FIND "-88"     REVDTDIR.WRK >> REVDTDIR.DIR
12 FIND "-87"     REVDTDIR.WRK >> REVDTDIR.DIR
13 FIND "-86"     REVDTDIR.WRK >> REVDTDIR.DIR
14 FIND "-85"     REVDTDIR.WRK >> REVDTDIR.DIR
15 FIND "-84"     REVDTDIR.WRK >> REVDTDIR.DIR
16 FIND "-83"     REVDTDIR.WRK >> REVDTDIR.DIR
17 FIND "-82"     REVDTDIR.WRK >> REVDTDIR.DIR
18 FIND "-81"     REVDTDIR.WRK >> REVDTDIR.DIR
19 FIND "-80"     REVDTDIR.WRK >> REVDTDIR.DIR
20 FIND /V "----------" REVDTDIR.DIR > REVDTDIR.WRK
21 COPY REVDTDIR.WRK REVDTDIR.DIR
22 DEL REVDTDIR.WRK
23 DIR ¦ FIND "File(s)" >> REVDTDIR.DIR
24 TYPE REVDTDIR.DIR ¦ MORE
```

20번행은 도스 FIND를 출력하기 위한 필터로 그 문자열이 발견되는 화일의 이름을 보이기 위한 것이다.

24번 행에서는 역순으로 디렉토리를 재지정하여 스크린 대 스크린으로 디스플레이하기 위해 MORE 필터로 보낸다.

다음의 리스트는 이 배치화일을 실행시킨 결과이다.

```
---------- REVDTDIR.DIR

 Volume in drive C has no label

 Directory of  C:\DOS\DOS3_3

MEN       TXT       70  4-25-89  11:48a
%PIPE     $$$     2977  4-25-89  11:36a
REVDTDIR  BAT      768  4-12-89  10:38p
WOMEN     TXT       71  4-12-89   9:29p
PC        TXT       20  3-21-89   3:50p
LIST      BAS     1103  3-05-89  11:49a
LIST1     BAS     1074  3-05-89  11:34a
LIST2     BAS     1082  3-05-89  11:18a
AZ        BAS      305  3-05-89   4:54p
Z         BAS        0  3-05-89   3:47p
TESTMSG   BAT      139  2-14-89   1:06p

..            <DIR>     8-12-88   3:26p
.             <DIR>     8-12-88   3:26p

GWBASIC   EXE    80592  7-24-87  12:00a
RECOVER   COM     4299  3-18-87  12:00p
-- More --
```

파이프와 필터는 매우 간단한 DOS의 기능이지만 디렉토리 구조와 트리구조 등과 함께 익힌다면 DOS에 대한 지식을 넓힐 수 있을 것이다. PC

□ 알림 □

6회에 걸쳐 독자여러분의 성원속에 연재된 본 강좌는 이번호를 마지막으로 끝마치게 되었음을 알려드립니다.

유용한 유틸리티 1
PC Tools

박승제/여주컴퓨터학원장

IBM-PC를 배우려는 분들은 좋튼 싫튼 MS-DOS를 익혀야만 한다. 그러나 DOS는 그것을 잘 다룰줄 아는 사람들에게는 아무런 문제가 없지만 초보자의 경우에는 그 많은 명령어를 일일이 외워야하고, 또 반드시 키보드로부터 입력을 해야하는 불편함과 번거로움이 뒤따르게 된다.

예를 들어 여러개의 화일을 디스크로 복사하려는 경우에 와일드카드 문자('*', '?')를 사용할 수 없는 경우라면 일일이 화일명을 지정하지 않으면 안될 것이다. 또, 사용자의 부주의로 인해 귀중한 화일이 지워졌을 경우에 MS-DOS는 이런 문제에 대해 어떠한 해결책도 제시해주지 못한다.

PC Tools은 Central Point Software사의 작품으로 MS-DOS의 환경을 지원해주는 유틸리티이다. 우리나라의 경우에도 PC를 조금이나마 다룰줄 아는 사람이라면 누구나 보유하고 있는 프로그램이다. PC Tools이 갖고 있는 기능을 열거하면 다음과 같다.

- 서브 디렉토리의 구조 표시 및 화일 디렉토리의 정렬(Directory)
- 화일 및 디스크의 복사(Copy)
- 복사된 화일 및 디스크를 원본과 비교(Compare)
- 화일 및 디스크상에서 특정한 문자열이 위치하는 부분을 탐색(Search)
- 화일명 및 디스크의 볼륨 레이블의 변경(Rename)
- 화일의 삭제(Delete)
- 화일 및 디스크의 결함상태 조사 및 복구(Verify)
- 화일의 내용을 디스크상에서 직접 수정(View / Edit)
- 지워진 화일의 복구(Undelete)
- 화일의 속성변경(Status)
- 디스크에 화일이 할당된 물리적인 위치의 표시(Mapping)
- 자신이 사용중인 컴퓨터 시스템에 대한 정보 제공(System Info.)
- 텍스트 화일을 프린터로 출력(Print)
- 디스크의 포매팅(Format)

PC Tools은 이름 그대로 PC(Personal Computer의 약어이나 요즘은 IBM-PC를 가르킨다)의 유용한 도구이다. 모든 도구들이 다 그렇듯이 잘못 사용하면 오히려 병이 되는 수가 있다.

PC Tools은 초기 버전인 1.01부터 현재는 4.20까지 보급되어 있으며, 버전이 높을수록 기능은 다양하다. 그러나 초보자의 경우에는 초기 버전인 1.01이 사용하기에 편리하며, 프로그램 사이즈가 작아 DOS디스크에 수록해 둘 수 있으므로 필자의 경우에도 릴리즈 1.01을 주로 이용한다. 이 컬럼이 초보자를 위한 것이므로 여기서도 R1.01에 대해서 설명한다.

PC Tools은 누구나 몇 시간만 조작해보면 그 기능을 쉽게 익힐 수 있다. PC Tools은 화일명이 PCTOOLS.EXE로 되어 있으며 DOS상태에서 PCTOOLS를 입력하면 바로 실행된다. PC Tools의 초기 화면은 **화면** 1과 같다. 여기서 커서 상하 키로 화면의 반전된 부분을 이동시켜 원하는 기능을 선택한다.

PC Tools의 매뉴얼은 한글로 번역되어 소개되고 있지만 이제 컴퓨터를 배우기 시작한 독자들을 위하여 간단하게나마 PC Tools의 기능과 사용법을 설명할까 한다.

DIRECTORY

[기능] **화일 디렉토리의 검색, 화일 속성(Attribute) 표시 및 화일 디렉토리의 정렬**

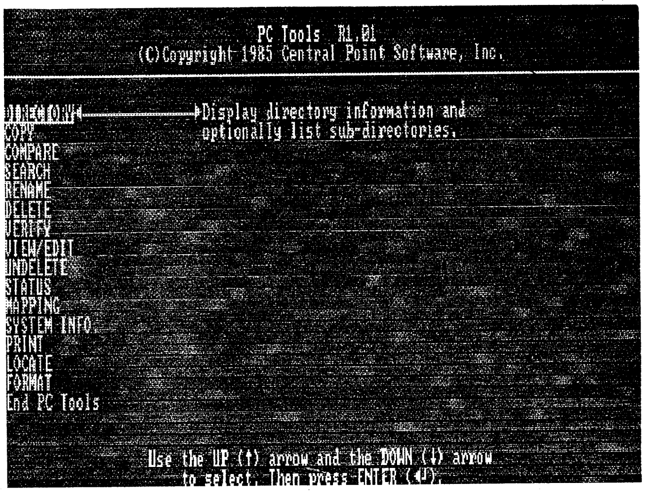
이 코맨드를 선택하면 드라이브 번호를 물어온다. 드라이브는 A, B처럼 입력한다. DOS에서 처럼 ':'을 뒤에 붙이지 않는다. (이것은 다른 코맨드의 경우에도 동일하다.)

PC Tools은 그 디스크에 서브디렉토리가 있는 경우에는 서브디렉토리를 그림으로 표시하여 준다. **화면** 2에서 해당 서브디렉토리의 이동은 커서 키에 의한다.

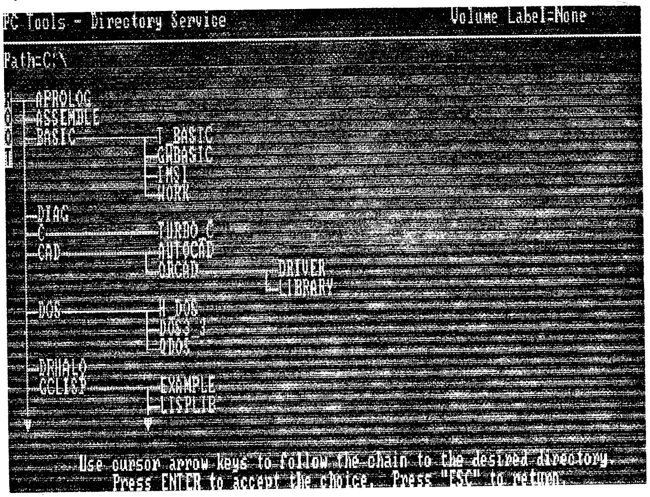
〈Esc〉키로 다시 원래의 초기화면상태로 돌아와서 ↲ 키를 눌러 디렉토리의 선택이 끝나면 화면에 디렉토리에 있는 화일명이 화면상에 나타난다. 이때 그 화일의 속성도 함께 표시되는데 PC Tools에서 표시하는 화일속성은 다음과 같다.

- **System** : 시스템용 화일이다. IBMBIO. COM, MSDOS.COM 등이 이 속성을 갖는다.
- **Read-Only** : 읽기 전용 화일이다. 이 화일은 DOS의 DEL코맨드로는 삭제되지 않는다. 물론 PC Tools에서는 삭제할 수 있다.
- **Hidden** : 숨겨진 화일로서 DOS의 DIR

〔화면 1〕 PC Tools의 초기 화면



〔화면 2〕 서브 디렉토리의 표시



명령으로는 디렉토리상에 나타나지 않는다. 그러나 이 화일은 DOS상에서 실행이 가능하다.

- **Archive** : DOS의 BACKUP 코맨드에 의해 백업된 적이 있는 화일이다.
- **Normal** : 보통 상태의 화일이다.

디스크에 화일이 많은 경우에는 화일의 일부분만 표시되므로 〈PgDn〉이나 〈PgUp〉키로 화면을 위 아래로 스크롤 시켜가며 볼 수 있다. 또, 〈Home〉키는 맨 처음 화일부터 보고자할 때, 〈End〉키는 맨 마지막 화일부터 보고자 할 때 사용된다.

이 코맨드에서 〈F7〉키는 디렉토리를 정렬하는 기능을 갖는다. 정렬은 화일 이름순으로 할 수도 있고, 확장자 순으로 할 수도 있으며, 화일이 생성된 날짜순 중에서 선택(F7=화일 이름순, F8=확장자 순, F9=사이즈 순, F10=날짜/시간순)할 수 있다.

일단 정렬이 되면 화면에 그 결과가 표시되며, 'U'(Update)를 입력하면 디스크상에 정렬된 상태로 기록하게 된다.

디렉토리에 화일이 많은 경우에는 〈F8〉을 누른 다음 와일드카드 문자('*', '?')를 지정하여 원하는 화일만 화면에 표시할 수도 있다.

COPY

[기능] **화일 및 디스크를 복사한다.**

이 코맨드는 여러개의 화일을 복사하거나 디스크를 통째로 복사하려는 경우에 아주 유용하다. 이 코맨드를 선택하면 화일이냐 디스크냐를 입력할 윈도우가 화면의 우측 중앙에 나타나는데 상하 이동키로 반전된 부분을 이동시킨 다음 ↵를 입력하면 복사가 이루어진다.

다음에는 소스 드라이브(Source Drive)와 타겟 드라이브(Target Drive)를 물어온다. 소스 드라이브는 원본이 있는 곳이며, 타겟 드라이브는 복사를 받을 드라이브를 말한다.

이때 화일을 복사하는 경우라면 화면에 소스 디스크의 화일 디렉토리가 나타난다. 커서키로 원하는 화일을 선택한 후 ↵를 치면 화일명 앞에 1이란 숫자가 나타나는데 이 숫자는 복사할 순서를 나타내기 때문에 1은 복사순서가 첫번째라는 뜻이다. 그 다음 복사할 것을 커서키로 계속 선택하면 선택한 순서대로 화일명 앞에 2, 3... 등의 숫자가 나타나는데 이 숫자대로 복사가 이루어진다. 화일을 잘못 선택한 경우에는 다시 ↵를 누르면 선택이 취소된다.

화일 선택이 모두 끝난 후 'G'(Go)를 누르면 화일의 복사가 시작된다. PC Tools은 복사를 하기에 앞서 타겟 디스크에 남아 있는 용량을 조사하여 충분한 공간이 있다고 판명되면 복사를 수행하고, 공간이 모자라는 경우에는 비프(Beep)음을 발생하고 화일의 선택이 모두 취소된다. 이 때는 남은 공간을 고려하여 몇개의 화일을 우선 선택하여 몇개는 다시 복사를 시도하여야 한다.

화일단위의 복사를 하는 경우에 플로피 드라이브에서 하드 드라이브로 복사할 때에는 시스템 화일은 복사하지 말아야 한다. 왜냐하면 시스템 화일이 하드디스크에 있는 COMMAND.COM과 버전이 맞지 않는 경우에는 컴퓨터가 하드에서 부팅되지 못하기 때문이다. 특히 한글 DOS를 사용하는 경우에 영문 DOS상의 시스템 화일을 하드 드라이브로 복사한 후에는 DOS가 제기능을 발휘하지 못하며, 심한 경우에는 하드를 다시 포맷하여야 한다. 행정 전상망용 한글 DOS를 사용하는 기종에서는 특히 주의하기 바란다.

디스크를 복사(Diskcopy)하는 경우에는 소스와 타겟 드라이브를 입력한 후 각각의 디스크를 드라이브에 넣고 ↵ 키를 치면 복사가 수행된다.

PC Tools은 먼저 소스 디스켓을 모두 메모리로 읽어들인 후 타겟 디스크에 쓰게 된다. 그러므로 메모리용량이 작은 경우에는 디스크의 일부분을 읽어들인 후 복사를 하고, 다시 읽어들이고 복사하는 과정을 몇차례 반복해야 한다. 디스크가 복사되는 과정은 화면상에 문자로 표시되는데 읽을 때는 'R'(Read)가 쓸 때는 'W'(Write)가 나타나게 된다.

디스크의 복사는 DOS의 DISKCOPY.COM과 동일한 기능이므로 프로텍트가 걸린 디스크는 복사가 안되는 수가 있다. 이런 디스크를 복사하려는 경우는 COPYII PC나 Copy Write라는 유틸리티를 이용해야 한다.

COMPARE

[기능] **화일이나 디스크를 비교한다.**

이 코맨드는 화일이나 디스크를 복사한 후에 원본과 복사본을 비교하려는 경우에 이용된다. 화일의 선택이나 디스크의 선택방법은 복사의 경우와 같다. 이 코맨드에서는 화일이 선택되었다는 것을 나타내기 위해 화일 이름 앞에 '--->'표를 붙여 준다. 잘못 선택된 화일은 다시 ↵ 를 입력하면 선택이 취소된다.

화일 선택이 끝나고 'G'(Go)를 입력하면 이름이 같은 화일끼리 비교를 할 것인가를 물어오는데 이때 'Y'나 'N'을 입력한다. 'N'을 입력할 경우에는 복사본의 화일 이름을 물어오게 되므로 이름을 지정하면 된다.

디스크의 비교시에는 디스크 복사와 동일한 방법으로 비교를 하면서 내용이 다른 부분이 발견되면 그 섹터를 화면에 표시하게 된다.

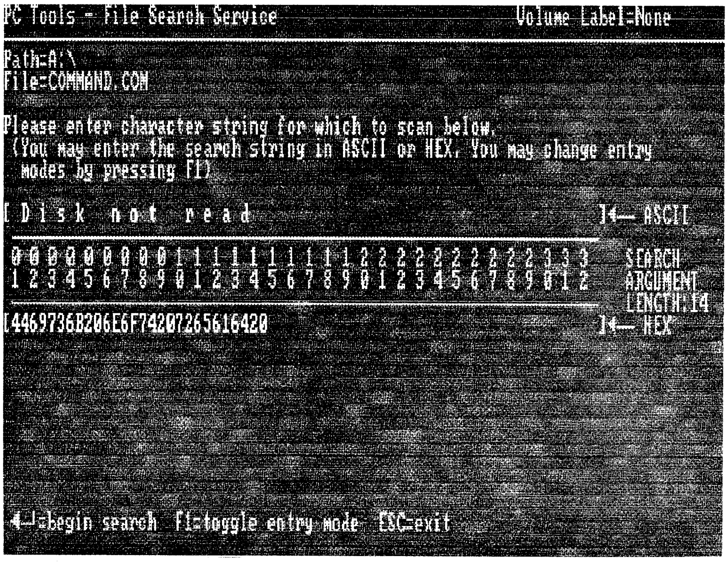
SEARCH

[기능] **화일 혹은 디스크상에서 특정한 문자열을 찾는다.**

이 코맨드는 화일이나 디스크상의 문자열을 찾아보거나 그 문자열을 다른 문자열로 대치하는 경우에 유용하다. 이를테면 COMMAND.COM의 경우 나타나는 에러 메시지인 'Abort, Retry, Ignore'를 '취소, 되풀이, 무시'등과 같이 한글로 바꾸로 싶은 경우 그 문자열의 위치를 빨리 찾아주게 된다.

우선 COMMAND.COM을 선택하면 찾을 문자열을 물어오는데 키보드에서 입력하면 된다(화면 3 참조). 키보드에서 입력할 수 없는 문자인 경우에는 그 문자의 아스키 코드를 16진수로 입력한다. 16진수를 입력할 경우에는 먼저 〈F1〉키를 입력하도록 하라. 〈F1〉는 토글(Toggle)되어 있으므로 다시 입력을 하면 문자를 입력할 수 있게 된다.

〔화면 3〕 SEARCH SERVICE 화면



화면 3에서 ↵ 를 입력하면 PC Tools은 그 문자열을 찾게 되고, 문자열이 발견되면 보기 1과 같은 메시지를 출력한다.

〔보기 1〕

```
Search argument found in relative sector 0000003
Press "E" to view/edit the sector or
      "G" to continue searching.
```

여기서 'E'를 선택하면 그 문자열이 있는 섹터를 수정할 수 있게 된다. 수정하는 방법은 다음에 나오는 'VIEW/EDIT'항을 참조하기 바란다. 'G'를 입력하면 그 다음 섹터부터 검색을 계속한다.

RENAME

[기능] **화일 및 디스크의 볼륨 레이블을 변경한다.**

이 코맨드를 선택하면 화일이냐 디스크냐고 물어오는데 메시지의 출력형태나 입력방법은 'COPY'의 경우와 동일하다. 디스크는 디스크의 볼륨 레이블을 의미한다.

다음에는 드라이브를 물어오며, 이때도 A, B... 등으로 입력한다. 화일이름을 변경하는 경우에는 먼저 변경할 화일을 모두 선택한다. 화일이 선택되

었다는 메시지는 'COMPARE'의 경우와 동일하다.

'G'(Go)를 입력하면 화일이름의 변경이 시작되는데 매번 새로운 이름을 입력한 다음에는 보기 2와 같은 메시지를 출력한 후 입력 대기상태가 된다.

메시지의 의미는 다음과 같다.

'Y' : 이름을 변경함

'N' : 새로운 이름을 다시 입력함

'B' : 이 화일은 변경하지 않고 다음 화일로 넘어감

ESC : 메인 메뉴로 돌아감

〔보기 2〕

```
DATE    .DAT is being renamed

Please enter the new file and extension names
(ESC will end)

                   Name=[TEST    ] ←── 새로운 이름을 입력함
                   Ext =[DAT]

DATE    .DAT is to be RENAMED to TEST    .DAT

Please confirm. "Y" to RENAME
              "N" to reenter
              "B" to bypass
              ESC to return
```

DELETE

[기능] 화일을 삭제한다.

화일 디렉토리상에서 삭제할 화일을 모두 선택한 다음 'G'에 의해 삭제작업을 수행한다. 삭제하고자 하는 화일을 여러개 선택한 후에는 'G'키를 누르면 보기 3과 같은 메시지가 출력된다.

이때 'Y'를 입력하면 모든 화일을 삭제하며 'N'을 입력하면 지울 화일을 보기 4와 같이 하나씩 화면에 표시하고, 그 화일을 삭제할 것인지를 물어오게 된다. 화일을 삭제할 것인지 물어 올 때 'Y'를 선택하면 삭제를 하며, 'N'을 선택하면 그 화일은 삭제를 하지 않고 그 다음 삭제하고자 했던 화일로 넘어간다.

〔보기 3〕

```
You have selected multiple files to delete.

If you want to delete all files without
individual confirmations, press "Y".
For individual confirmations, press "N".
Press ESC to Exit
```

〔보기 4〕

```
CHECK   .EXE is to be DELETED

Please confirm. "Y" to DELETE
                  "N" to bypass
                   Press ESC to Exit
```

VERIFY

[기능] 화일이나 디스크상의 결함을 체크한다.

디스크나 화일이 물리적으로 손상되었는지 여부를 체크한다. 화일이나 디스크가 손상되었을 경우에는 섹터번호를 표시하며 베리파이(Verify)는 중단된다. 이때 ↵ 키를 치면 중단된 다음부터 베리파이가 계속된다.

VIEW/EDIT

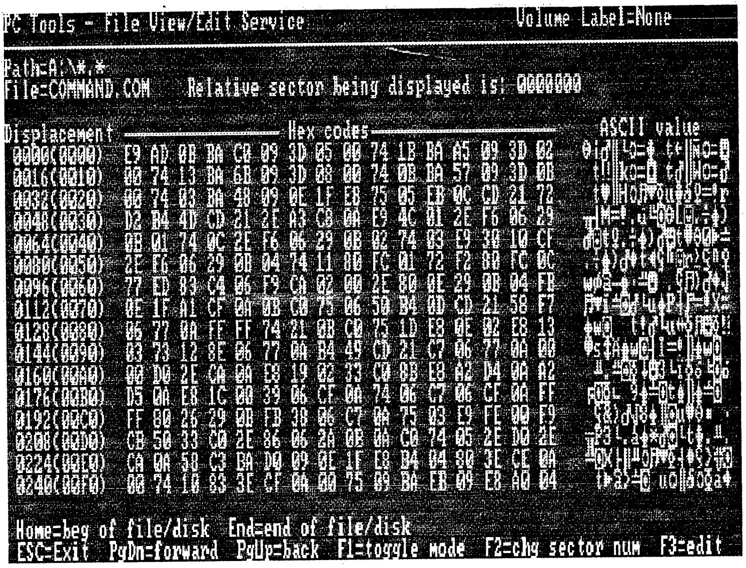
[기능] 화일이나 디스크를 섹터별로 검색, 수정을 한다.

화일의 일부분을 디스크상에서 직접 수정하려는 경우에 유용한 코맨드이다. 여기서 수정이라는 의미는 대치하는 것이지 편집기능은 아니다. 즉, 새로운 내용을 디스크에 끼워 넣거나 일부분을 삭제한 후 뒷부분을 앞으로 당기는 기능은 없다.

이 코맨드는 맨 처음의 섹터부터 1섹터분을 화면에 16진수와 아스키문자로 표시한다. 화면 4에서 〈PgDn〉키는 다음 섹터를 검색하려는 경우에 이용되며, 〈PgUp〉키는 앞의 섹터를 검색하는데 사용된다. 또, 〈Home〉키는 화일이나 디스크의 선두 섹터로 이동할 때 사용되며, 〈End〉키는 맨 끝의 섹터로 이동하는데 사용된다.

이 코맨드에서 〈F2〉키는 특정한 섹터번호를 지정하여 그 섹터를 즉시 화면에 표시하는 경우에 이용된다. 화면상의 섹터를 수정하려는 경우에는 〈F3〉키를 누른다. 그러면 화면의 16진수 부분에 커서가 나타나게 된다. 수정된 내용을 디스크에 수록할 경우에는 〈F5〉

〔화면 4〕 VIEW/EDIT SERVICE 화면



키를 입력하면 된다. 또 〈F6〉키는 수정한 것을 취소하는 경우에 이용된다.

〈F1〉키는 16진수로 표시된 부분과 문자로 표시된 부분을 이동하는 경우에 이용되며, 키보드에서 입력이 가능한 문자라면 키보드상에서 수정하는 것이 좋다. 〈F2〉키는 텍스트 화일을 행단위로 표시해준다.

UNDELETE

[기능] 지워진 화일을 복구한다.

DOS의 'DEL' 명령 등으로 화일을 삭제하는 경우 실제는 화일이름의 첫 문자를 E5H로 바꾸고, 그 화일의 선두 클러스터(Cluster)만 삭제되었을 뿐 그 내용은 고스란히 디스크상에 남아 있게 된다. PC Tools는 DOS의 이런 성질을 이용해서 지워진 화일을 원상대로 복구하게 된다.

단, 모든 화일이 다 복구되는 것은 아니다. 화일을 삭제한 후 그 디스크에 새로운 화일을 만들거나 복사 등을 한 적이 없는 디스크는 대부분 제대로 복구되나 그렇지 않을 경우에는 일부분밖에 복구되지 못한다. 지워진 화일이 텍스트 화일이거나 데이터 화일이라면 일부나마 복구해도 별무리는 없겠지만 프로그램 화일인 경우에는 일부분만 복구할 경우 제대로 수행되지 않기 때문에 이점을 유념해야 한다.

이 코맨드를 선택하면 지워진 화일 목록만 화면에 표시해주는데 첫문자는 모두 '?'로 되어 있다. 그 이유는 앞에서 설명한 바와 같이 화일이름의 첫 문자는 모두 지워졌으므로 PC Tools로서도 지워진 문자가 어떤 문자인지 알 수가 없기 때문이다. 복구하려는 화일을 선택한 다음 'G'를 입력하면 다음과 같은 메시지를 출력하는데 이것은 자동적으로 복구하겠느냐 수동으로 복구하겠느냐라는 의미이다.

```
Use function keys to make selection
   F1 - Automatic selection of clusters
   F2 - Manual selection of clusters
ESC - Return
```

대부분은 〈F1〉키로 복구가 된다. 복구하려는 화일이 프로그램인 경우에는 〈F1〉키로 복구를 시도해 복구가 안되면 포기를 하는 편이 낫다. 〈F2〉를 선택한 경우에는 1클러스터(2섹터)씩 화면에 표시한 후 보기 5와 같은 메시지가 출력된다.

〈F1〉은 이 클러스터를 포함한다는 의미이고, 〈F2〉는 포함하지 않고 건너뛰며, 〈F3〉은 〈F1〉으로 선택한 클러스터들을 한개의 화일로 묶어 디스크에 저장한다는 의미이다.

STATUS

[기능] 화일 속성을 변경한다.

임의의 화일을 선택하여 그 화일의 속성 및 생성 날짜 등을 변경한다. 변경항목은 화면 5처럼 되어 있으며 커서키로 원하는 항목으로 이동한 다음 키를 치면 속성이 반대로 된다. 또 화일의 생성 날짜와 시간들은 키를 누른후 직접 수치를 입력하면 된다. 'A'키를 입력하면 변경된 상태를 디스크에 저장하게 된다.

MAPPING

[기능] 디스크에 화일이 할당되어 있는 모습을 그림으로 표시한다.

이 코맨드는 화일이 디스크를 점유하고 있는 비율, 화일이 디스크에 저장된 물리적인 위치, 시스템 화일이나 히든화일, 불량 섹터의 위치 등을 그림으로 표시한다. 초기의 화면은 디스크의 점유율을 보여주며, 여기서 'F'를 입력하면 화일단위로 표시를 하게 된다. 'F'를 입력하면 화일목록이 출력되는데 아무 화일이나 선택한 다음 'G'를 입력하면 그 화일이 저장된 위치가 표시된다. 이때 '->'키는 다음 화일을, '<-'키는 앞의 화일을 볼 수가 있다.

〔보기 5〕

```
F1=Add to file and display next cluster, F2=Skip to next cluster
F3=Save file, ESC=Exit
```

'D'를 입력하면 다시 디스크 전체를 볼 수 있게 된다.

SYSTEM INFO.

[기능] 현재 사용중인 컴퓨터 시스템에 대한 정보를 출력한다.

여기서 정보라는 것은 드라이브를 모두 몇대까지 연결할 수 있으며, 사용 가능한 메모리는 얼마인가 하는 것들이다. 화면 6에 출력화면을 보였으니 참고하기 바란다.

PRINT

[기능] 화일을 프린터로 출력한다.

이 코맨드에서 프린터로 출력하는 경우는 텍스트 화일인 경우에는 텍스트로, 바이너리 화일인 경우에는 16진수 덤프리스트를 출력한다. 출력할 화일을 미리 선택한 다음에 'G'를 입력하면 화면 7과 같은 메시지가 나오게 되는데, 'P'는 텍스트 포맷으로, 'D'는 16진수 덤프리스트를 출력한다는 의미이다.

LOCATE

[기능] 와일드카드 문자('*', '?')를 지정하여 특정한 화일만 표시되도록 한다.

PC Tools에서는 디폴트(Default)로 '*.*'이 지정되어 있으므로 항상 모든 화일이 표시되게 된다. 특별한 경우가 아니면 이 코맨드는 선택하지 않아도 된다.

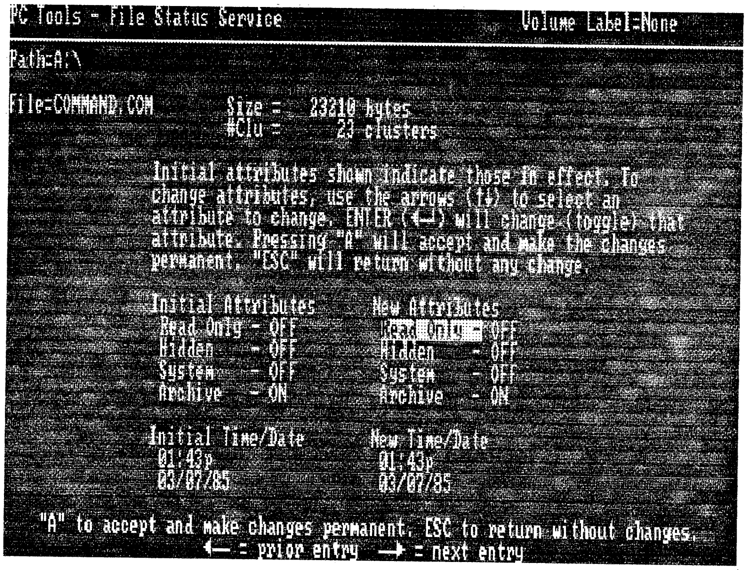
FORMAT

[기능] 디스크를 포맷한다.

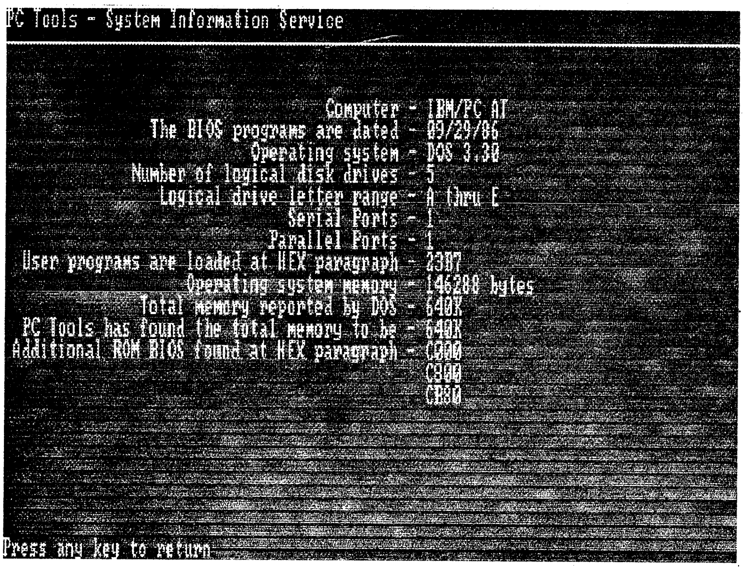
PC Tools에서는 여러 형태의 포맷이 가능하다. 초기의 화면은 보기 6과 같다.

보통은 양면 배밀도로 포맷하게 되므로 360K를 선택하면 된다. 320K는 양면 배밀도 디스크에서 트랙당 8섹터

〔화면 5〕 STATUS. SERVICE 화면



〔화면 6〕 SYSTEM INFO. SERVICE 화면



〔화면 7〕 PRINT 서비스 화면

```
File=FORMAT.COM

Please specify your print options as follows:

 "P" - print as a standard text file (file contains control characters
        as necessary)
 "D" - each sector DUMPED in ASCII and HEX
 "N" - skip this file and go to next one selected
 ESC - return
```

〔보기 6〕

```
Drive A is about to be formated.

Choose the formating desired
and press ENTER to begin.

360K-> double-sided, 9 sectors per track
320K
180K
160K
```

로 포맷하는 경우이며, 180K, 160K는 단면 9트랙, 단면 8트랙으로 포맷하는 경우이다.

포맷이 끝나면 시스템을 전송할 것이냐는 메시지가 출력되는데 'Y'를 입력하면 시스템이 전송될 영역을 마련하게 된다. 여기서는 영역만 마련할 뿐 실제의 전송은 수행되지 않는다. 그러므로 'COPY'코맨드를 선택해서 'IBMBIO. COM', 'IBMDOS.COM', 'COMMAND.COM' 등의 3화일을 복사하여야 한다.

끝으로

이상으로 PC Tools에 대해서 알아보았다. PC Tools는 버전이 많이 향상되어 3.0이상에서는 램상주도 허용하고 있지만 아무래도 초보자의 경우는 메뉴 선택형이 사용하기에 편리하다.

PC Tools의 기능을 완전히 익힌 독자라면 새로운 버전인 4.20을 사용해 볼 것을 권한다. 그렇다고 4.20이 다 좋다는 것은 아니다. 물론 프로그램이란 버전이 향상될수록 기능이 다양하겠지만 그만큼 사용법이 복잡하게 된다. 또한 프로그램의 크기가 증가하는 관계로 디스크에서 로드할 때 시간이 걸리게 된다.

물론 하드 디스크를 사용하는 경우라면 별로 상관이 없겠으나 플로피 디스크에서 불러들이는 경우에는 기다리는 시간이 퍽 지루하게 느껴진다. 그런 까닭에 필자의 경우에도 초기의 버전인 1.01을 주로 이용하며, 1.01에서 불편한 부분만 4.20을 사용한다. 아무쪼록 이 컬럼이 독자들의 컴퓨터 생활에 많은 도움이 되기를 바란다.

〈용어해설〉

● **토글스위치(Toggle Switch)**
손으로 직선적인 상하 왕복동작을 하여 전기회로를 개폐하는 레버형태 스위치의 일종으로 수동속도 여하에 관계없이 내부의 용수철로 접점이 급속히 이동하여 개폐된다. 따라서 그 조작레버가 동작상태를 표시하고 있는 것을 이용하여 컴퓨터의 정보입력 등에 사용된다.

● **섹터(Sector)**
1개의 트랙을 몇개로 분할한 블록으로 레코드라고도 불리며 리드/라이트는 섹터단위로 이루어진다.

● **유틸리티(Utility)**
유틸리티프로그램이라고도 하는데 소트루틴(Sort Routine), 프린터출력(Printout) 프로그램 및 화일변환 프로그램 등과 같이 컴퓨터의 작동을 보조하는 프로그램이다. 일반적으로 이 유틸리티 프로그램은 하우스키핑(Housekeeping)을 실행하거나 실제의 자료처리에 관여한다.

● **하우스키핑(Houskeeping)**
컴퓨터의 처리를 능률적으로 행하기 위한 관리기능이다. 상위 오프레이션이지만 이것이 직접 프로그램의 문제를 처리하는 것은 아니다.

● **클러스터화(Clustering)**
특정관리의 일부분에서 각 패턴의 특정분포가 밀집되어 있는 것을 말하며, 그와 같은 당면을 이용하여 각 패턴을 동일한 종류의 성질을 가진 것끼리 분류하는 조작을 말한다.

● **덤프(Dump)**
기억장치의 데이터를 기록매체에 라이트(Write)하는 것이다. 말하자면 어떤 기억장치 전체 또는 일부의 내용을 복사하는 것이다. 보통 내부 기억장치의 내용을 외부기억장치에 복사하는 것을 말한다. 출력형식을 2진, 8진 16진의 ASCII코드 등을 사용한다. 또 프로그램 실행중에 에러가 생겼을 때 메모리의 내용이나 기억메체상의 데이터를 확인하거나 매체변환을 하는 경우에 사용되는 예도 있다.

● **버그(Bug)**
하드웨어나 소프트웨어의 착오, 즉 오류 또는 오동작을 말하며 특히 프로그램의 에러를 가리킨다. 원래는 벌레라는 뜻이다.

● **벤치마크(Benchmark)**
하드웨어나 소프트웨어 또는 양쪽의 성능을 평가하기 위해 사용되는 문제이다. 예를 들면 표준이 되는 문제를 설정하여 평가대상이 되는 컴퓨터를 상대로 처리시간을 측정하여 컴퓨터를 상대로 처리시간을 측정하여 성능평가를 한다. 표준문제로는 일반적으로 화일처리, 직접 액세스 화일의 랜덤처리, 매트릭스의 반전 등이 있다.

● **인텔리전트 터미널(Intelligent terminal)**
단말기능으로서 고성능의 마이크로프로세서 등을 내장한 것을 말함, 컴퓨터 온라인 데이터 처리기능과 데이터 통신기능을 지녀 다기능 단말장치라고도 불린다. 이것의 특징은 프로그램 처리기능 및 데이터 처리기능, 화일 처리기능과 플로피디스크의 사용, 온라인 단말기로서의 통신기능 등을 들 수 있다.

● **온라인 실시간 시스템**
컴퓨터에 작업이 입력되면 그때마다 처리하는 방식.

컴퓨터 수강생 모집

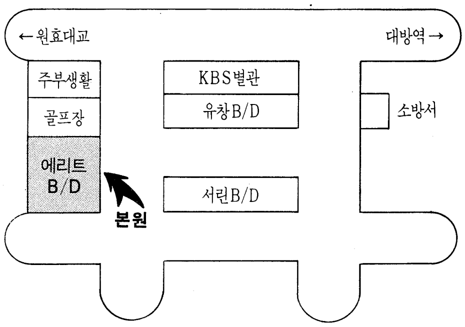
최첨단 기술을 가르치는 컴퓨터 분야는 전통만이 능사는 아닙니다.
자격과 능력을 갖춘 강사가 몇분이나 계신가를 확인하셔야 합니다.

그런면에서 저희 대기 컴퓨터 학원은 강사진 전원이 이공과대학 전산학과 전자공학과 정보처리학과 출신자들로 학위는 물론 기사1급 자격 소유자들로 구성되어 있읍니다.

또한 매월 20시간 이상의 실습을 통하여 컴퓨터가 철학처럼 느껴지지 않게 산교육을 시키고 있읍니다.

○교 육 과 정○

정규과정 (7개월)	EDPS개론	1개월
	FORTRAN언어	2개월
	COBOL언어	〃
	ASSEMBLY (8086계열) 언어	〃
특별과정 (6개월)	BASIC언어	1개월
	File처리	〃
	PASCAL	2개월
	C	〃
디자인과정 (3개월)	AutoCAD	2개월
	Graphic응용	1개월
전산기기 (Hardware) 과정 (5개월)	Digital	1개월
	전산기구조	〃
	processor분석	〃
	자동제어	〃
	interface 및 sensor	〃
사무자동화 (OA) 과정 (5개월)	OS	1개월
	dBASE Ⅳ	2개월
	Spread sheet	1개월
	Wordprocessor	〃
통계처리 (2개월)	SAS	〃
	응용통계	〃



IBM컴퓨터 전문교육

관인 대기 컴퓨터 학원

서울시 영등포구 여의도동 44-32 에리트빌딩 5층
TEL : 780-6059, FAX : 782-3882

둘리가 본 컴퓨터세계 4

컴퓨터와 컴퓨터사이

이만희/한국교육개발원 컴퓨터연구센터

사랑이 깨지지 않도록

컴퓨터 디스크가 사용하지 못하게 되었을 때 흔히 디스크가 깨졌다고 말한다. 그러나 플로피디스크는 부드러운 플라스틱 원판에 외부는 다시 보호카바(자켓)로 잘 싸여있어 일부러 떨어뜨려도 깨지기는 커녕 쉽게 부러지지도 않는다. 그런데도 대부분의 사람들이 깨졌다고 말하는 이유는 디스크의 겉모양이 부서지지는 않지만 아차하는 순간 비싼 유리잔이 산산조각으로 깨지듯 기록된 내용이 순식간에 사라지기 때문이다.

컴퓨터를 다룰 때 가장 조심해야 하고 또 흔히 일어나는 사고가 디스크에 담긴 내용이 깨지는 것이다. 때로는 한장의 디스크에 컴퓨터 기계값보다도 값비싼 자료가 담겨 있을 수도 있다. 이 디스크의 구조 및 깨지는 이유에 대해 알아보기로 하자.

(1) 디스크의 구조

디스크는 음악용 레코드 판과 같은 모습으로 재질은 부드러운 플라스틱에 자성물질을 칠하고 다시 비닐층을 얇게 입혀 표면을 보호한다. 디스크를 컴퓨터 장치(디스크 드라이브)안에 넣으면 레코드 판처럼 빙글 빙글 돌아가며 전축바늘 대신 녹음기의 마그네틱 헤드와 같은 전자석 바늘로 자기의 극성을 이용해 자료를 기록한다.

(2) 트랙과 섹터

운동장에서 하얀 선으로 트랙을 구별하듯이 컴퓨터 디스크에서는 전자석으로 원형의 판 위에 자석선을 그어 트랙을 만들어 사용한다. 이 동그란 원형의 자석트랙은 표준 디스크가 40라인이고 보다 정밀한 고밀도 디스크의 경우는 80라인을 그어 사용한다. 트랙을 보다 많이 그을수록 많은 용량을 기억할 수 있지만 마그네틱 헤드를 매우 정밀하게 만들어야 하고 조절이 어려워진다.

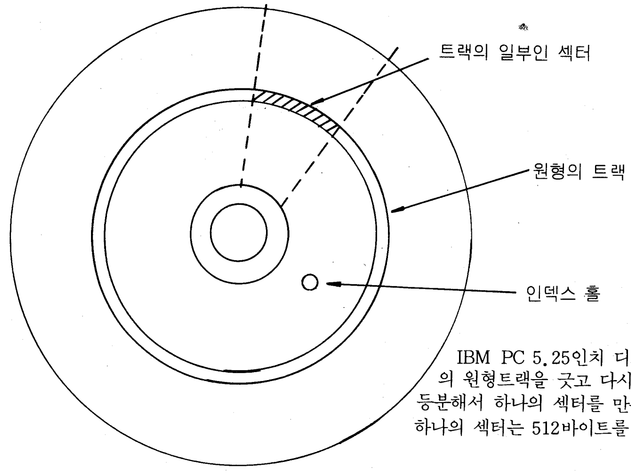
이 트랙은 다시 몇개의 부분으로 나누어 섹터라 부르고 있는데 릴레이 경기에서 한 사람이 달리는 구간이 섹터라고 생각하면 된다. 하나의 섹터는 512바이트를 기록하므로 한글로 치면 200자 원고지 한장이 약간 넘는다. IBM PC에 사용되는 표준디스크는 40트랙×9섹터×2(아래위 양면)이므로 총 720섹터가 되어 원고지 720장 즉, 보통 두께의 책한권 정도가 기록된다고 할 수 있고 360 킬로 바이트가 된다.

디스크에는 조그만 구멍이 하나 뚫려 있는데 이것을 인덱스 홀(Index hole)이라고 부르며, 이 구멍을 기준점으로 해서 트랙과 섹터 위치를 찾아간다. 디스크에 자료가 기록된 위치를 가르킬 때는 몇번 트랙의 몇번 섹터라 부른다(그림 1 참조).

(3) 디스크가 깨지는 이유

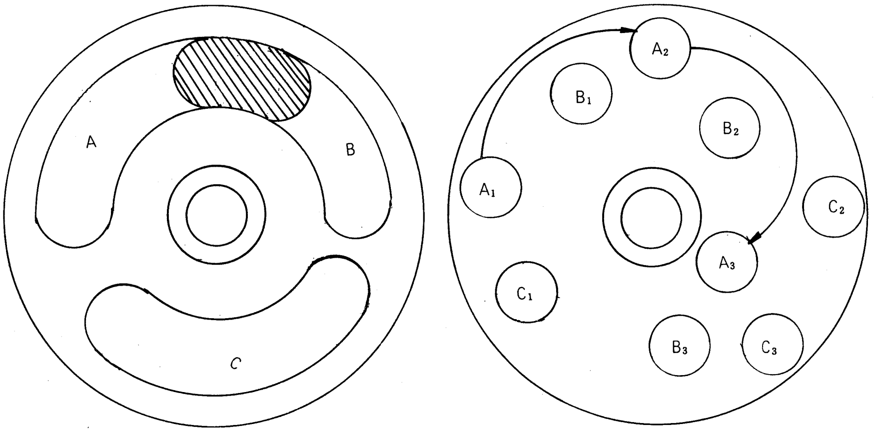
디스크의 기록방식은 보통 우리가 노트에 손으로 적는 방식과 다른 방법을 사용한다. 우리가 한권의 노트에 몇가지 과목을 함께 정리한다면 그중 한 과목의 분량이 점점 커져 예상했던 페이지를 넘어가면 다른 과목을 정리하는 부분에 걸려서 더 이상 계속 기록하기가 어려워진다. 그러나 한장의 컴

〔그림 1〕 디스크의 구조



IBM PC 5.25인치 디스크는 한면에 40개의 원형트랙을 긋고 다시 하나의 트랙을 9등분해서 하나의 섹터를 만든다.
하나의 섹터는 512바이트를 기록한다.

〔그림 2〕 화일의 기록



A라는 이름의 화일이 자료가 증가하여 커지면 B가 기록된 위치와 겹쳐서 기록할 수 없게 된다.

(a) 화일을 하나의 연속된 덩어리로 기록할 경우

하나의 섹터에 기록이 다 차면 비어 있는 섹터를 찾아서 기록한다. 옮겨가기 전에 옮겨가는 섹터의 위치를 기록한다.

(b) 화일을 섹터별로 기록할 경우

퓨터 디스크에는 여러개의 다른 화일을 함께 기록하면서도 화일의 크기가 커지거나 줄더라도 이상없이 기록할 수 있다.

이것은 하나의 화일을 여러 부분으로 나누어 섹터별로 기록하기 때문이다. 즉 하나의 섹터에 기록이 끝나고 다음 위치의 섹터가 다른 화일에서 사용되고 있다면 몇개의 자리를 건너 뛰어 사용되지 않는 섹터로 옮겨간다. 단 다른 섹터로 옮겨가기 전에 건너 뛰어가는 섹터의 위치를 마지막에 표시한다(그림 2 참조).

이 방식이 좀 복잡해 보이지만 여러개의 화일을 한장의 디스크에 함께 기록할 때 각각의 화일 크기가 다르더라도 문제없이 디스크를 사용할 수 있는 커다란 장점이 있다.

문제는 하나의 화일을 여러개의 부분으로 나누어 기록하기 때문에 연결하는 고리가 깨지면 다른 화일과 섞여서 다음 부분을 찾아갈 수 없다. 눈으로 보기에 겉모습은 전혀 이상이 없어 보이지만 디스크의 한부분만 지워져도 함께 들어 있는 여러개 화일의 각 부분이 기록된 순서가 뒤죽박죽되어 그 디스크 전체가 알아 볼 수 없게 된다. 다시 말해 마치 조금만 깨져도 전체가 산산조각이 나는 유리잔과 같이 기억된 내용이 사라지기 쉽게 되어 있지만 여러개의 화일을 하나의 디스크에 함께 기록하기 위해서는 어쩔 수가 없다.

디스크의 자료가 파괴되면 기록한 내용은 찾기는 어렵지만 디스크가 외형적으로 부서진 것이 아니라면 내용이 깨진 디스크도 새로 포매팅을 해서 다시 사용할 수도 있다.

(4) 디스크가 깨지는 경우

- 전화기나 TV와 같은 전자제품 위에 디스크를 가까이 두면 발생하는 전파 및 자장에 의해 디스크의 기록이 산산조각 나는 수가 있다.
- 사무실에서 가끔 생기는 일로 실핀이나 클립을 자석에 붙여 놓았다가 사용하는 경우가 많다. 이때 실핀이나 클립도 자석으로 변하게 되는데 이것이 디스크 위에 닿으면 디스크의 기록은 간단히 지워져 버린다.
- 책이나 무거운 물건 밑에 디스크가 눌리면 디스크의 외형이 구부러져 다시 사용할 수 없게 된다.
- 네모난 보호카바 가운데 디스크표면 일부분이 들어나 보이는데 이 부분에 티끌이나 담배재가 들어 가거나 손자국 등이 생기면 디스크에 손상이 올 수 있다.
- 음식물이나 과자를 먹으면서 디스크를 만지면 그 부스러기가 들어가서 디스크에 치명적인 피해를 주기도 한다.
- 커피 및 음료수 : 정말로 주의할 점은 커피같은 음료수이다. 커피가 디스크 위에 엎질러지면 디스크 보호카바와 디스크 사이에 스며 들어가 그 디스크는 재생불능이 되고 입력된 자료는 영원히 다시 찾을 수 없다. 음료수는 디스크 뿐 아니라 컴퓨터 본체나 키보드에도 큰 문제가 된다. 음료수를 컴퓨터 위에 놓고 있다가 실수로 넘어뜨려 키보드나 컴퓨터에 들어가면 큰 고장이 난다. 더우기 컴퓨터 회사에서 수리를 해준다 해도 사용자 실수가 분명하므로 수리비를 전부 부담해야 한다. 컴퓨터나 디스크를 놓는 책상에는 커피나 음료수를 가까이 하지 않는 것을 철칙으로 삼아야 한다.

컴퓨터와 컴퓨터사이

2차 대전시 독일에는 "오토-반"이라는 세계 최초의 고속도로망이 있었다. 히틀러는 이 고속도로망을 통해 연합군이 예상하지 못한 신속한 속도로 적시 적소에 군대를 옮기는 이동전법으로 커다란 전과를 올릴 수 있었다. 군대는 필요한 시간에 필요한 장소로 나타나면 되는 것이다. 마찬가지로 컴퓨터로 일을 하기 위해서는 필요한 정보를 보관하고 있는 컴퓨터와 연결하여 그 정보를 사용하면 되는 것이다. 미련스럽게 항상 모든 정보를 자신의 컴퓨터에 유지하고 있을 필요가 없다.

물자를 이동할 때 "교통"이 필요하듯이 정보를 전달하기 위해서는 "통신"이 필요하다. 따라서 컴퓨터와 컴퓨터 사이에 정보를 전달하는 통신망 즉, "컴퓨터 네트워크"를 만들게 되었다. 컴퓨터 네트워크는 고속도로의 차원이 아닌 말 그대로 "번개속도"로 많은 정

보를 전달한다. 컴퓨터 네트워크는 멀리 지구의 반대편에 떨어져 있는 컴퓨터와 연결하여 함께 정보를 이용하도록 하여 컴퓨터의 위력을 증가시키고 있다. 반대로 다른 컴퓨터와 사이가 나쁜 컴퓨터 즉, 다른 컴퓨터와 통신이 불편한 컴퓨터는 기능이 아무리 좋아도 훌륭한 컴퓨터가 될 수 없다.

(1) 컴퓨터 통신으로 할 수 있는 일

●자료검색

요즘 유행하는 증권투자도 컴퓨터 통신을 이용하면 집에서 쉽게 할 수 있다. 시시각각 변화하는 증권가격을 계속 입력하여 컴퓨터 네트워크로 연결된 회원에게 일정 비용을 받고 증권 거래가격을 알 수 있도록 해주는 회사가 생겼다. 이런 컴퓨터 네트워크가 생기면서 보다 다양한 정보서비스도 제공되기 시작했다. 문화 행사나 영화안내, 부동산정보, 관광안내 등의 계속 변하는 자료가 입력되어 손님들의 이용을 기다리고 있다.

최근에는 컴퓨터를 이용해서 공부를 하는 CAI 프로그램도 대형 컴퓨터에 모아서 입력을 해놓고 컴퓨터 네트워크를 통해 필요한 사람이 집에서 원하는 과목의 CAI 프로그램을 불러 공부할 수 있도록 연구하고 있다.

●전자우편

최근 조사에 의하면 업무상 전화를 해서 원하는 사람과 통화할 수 있는 비율이 불과 3분의 1로 세번 걸어야 한번 연결이 된다고 한다. 편지로 보내면 확실히 전달되지만 며칠씩 걸린다. 이때 컴퓨터 네트워크를 이용하면 편지를 순식간에 보낼 수 있다.
데이타 통신(주)에서는 전자사서함 업무를 하고 있는데 원하는 사람이 가입하면 데이타 통신에서 운영하는 중계용 컴퓨터 기억장치에 자신의 사서함을 만들 수 있다.

다른 사람이 그 사서함에 전자우편(Electronic Mail)을 보내면 자동적으로 컴퓨터에 보관되어 그가 네트워크를 통해 연결하면 자신의 사서함에 쌓여 있는 전자우편을 받아 볼 수 있다. 전자우편의 장점은 편지를 받아 보는 사람이 한 곳에 고정되어 있지 않고 여러 장소에서 통신망을 통해 컴퓨터를 연결하여 전자우편을 받아 볼 수 있다는 것이다.

예컨대 외국 여러나라로 여행중인 사장님이 묵고 있는 호텔방에서 편리한 시간에 컴퓨터를 네트워크에 연결하면 회사에서 보낸 소식이 담긴 전자우편을 받아 볼 수 있다.

필자는 몇년 전에 컴퓨터 동우회를 결성하고 회지를 발간한 적이 있었는데 회지를 매달 작성하기가 너무 힘들었다. 그런데 미국에서는 뉴스룸이라는 신문편집 프로그램을 사용하여 신문을 만드는데 회원들이 각자 집에서 신문의 일부분을 작성하고 그것을 네트워크로 모아서 완성한다는 말을 듣고 크게 부러워한 적이 있다.

전자사서함을 이용한다면 한밤중에 기사를 만들어 보내는 회원도 있고 해변 휴양지에서 신문을 만드는 발행인도 있을 것이다. 아무리 멀리 떨어진 장소에서라도 각자 자기의 맡은 일을 하면서 컴퓨터 네트워크를 통하여 편리하게 마무리 지을 수 있다.

●BBS(Bulletin Board System)

전자사서함과 분담작업의 기능을 이용해서 서로 모르는 사람들이 HAM(아마추어무선)처럼 하나의 클럽을 이루어 컴퓨터로 서로 교신할 수 있다.

BBS는 일종의 공개된 컴퓨터 게시판과 같은 것으로 그 지역 컴퓨터 사용자들의 지방신문 역할을 한다. 그리고 아마추어 프로그래머들이 자신이 개발한 프로그램을 발표하고 무료로 소프트웨어를 주고 받는 수단으로도 이용된다. 그러나 요즘 공포의 대상인 컴퓨터 바이러스가 이 BBS를 통해 전달되는 무료 소프트웨어 속에 몰래 숨어서 전염되어 수많은 사용자가 고통을 받기고 하였으니 조심해야 한다.

●주변기기의 공유 및 터미널로 사용

소형 퍼스널컴퓨터가 보급되고 있지만 대형 컴퓨터가 편리한 점도 있다. 수백만건의 자료를 갖고 통계처리를 하거나 이미 축적된 방대한 자료를 이용하는데는 아무래도 대형 컴퓨터쪽이 편하다. 이때 퍼스널컴퓨터를 컴퓨터 네트워크로 연결시키면 자기방에서 대형 컴퓨터를 이용할 수 있다.

단순한 터미널 기능 뿐 아니라 보다 편리한 인텔리전트터미널 역할을 한다. 즉, 대형 컴퓨터에 항상 연결할 필요없이 입력이 필요한 자료는 우선 퍼스널컴퓨터에 입력한 다음 네트워크에 연결한 후 자료를 대형 컴퓨터로 전송하여 자료의 처리는 대형 컴퓨터에서 하는 방식이다. 이렇게 하면 복잡한 전산실에 가지 않고도 대형컴퓨터의 장점만을 이용할 수 있다.

(2) 컴퓨터 통신에 필요한 것

●통신포트

독립된 퍼스널컴퓨터 내부에서 신호가 밖으로 나오려면 출구가 있어야 한다. 이 출구를 통신포트라고 부른다. 인터페이스라고도 부르는데 컴퓨터와 다른 장치를 연결시키기 때문이다.

통신포트로 제일 유명한 것은 RS-232C 포트가 있다. 본래는 대형 컴퓨터와 터미널을 연결하는데 사용했으나 퍼스널컴퓨터에서 RS-232C포트를 갖고 있으면 대형컴퓨터의 터미널로 쓸 수 있고 다른 컴퓨터와 자료를 주고 받는 창구로도 연결할 수도 있다. 또 RS-232C 포트에 프린터나 마우스 등 여러가지 입출력 장치를 연결할 수도 있다. RS-232C 방식이 많이 쓰이기는 하지만 최근에는 SCSI 포트라는 것도 보급되고 있는데 이것은 하나의 포트에 통신장치를 비롯해 프린터, 광디스크 등 7개까지의 장치를 병렬로 연결하는 것이 가능하다.

●통신선로

신호가 컴퓨터의 포트를 벗어나면 그 신호가 지나갈 수 있는 선로가 필

요하다. 가까운 거리라면 선을 직접 연결할 수 있지만 먼 지방에 떨어져 있으면 통신선로를 연결할 수 없다. 이때 간단한 해결방법은 전화선을 이용하는 것이다. 전화는 이세상 어느 곳이라도 연결되어 있다. 심지어 달리는 자동차 속에서도 전화는 연결된다. 물론 전화선로는 사람의 목소리를 전달하기에 적합한 것이므로 컴퓨터를 사용하기 위해서는 맞지 않는 부분이 있어서 모뎀이라는 장치가 필요하다.

●모뎀(Modem)

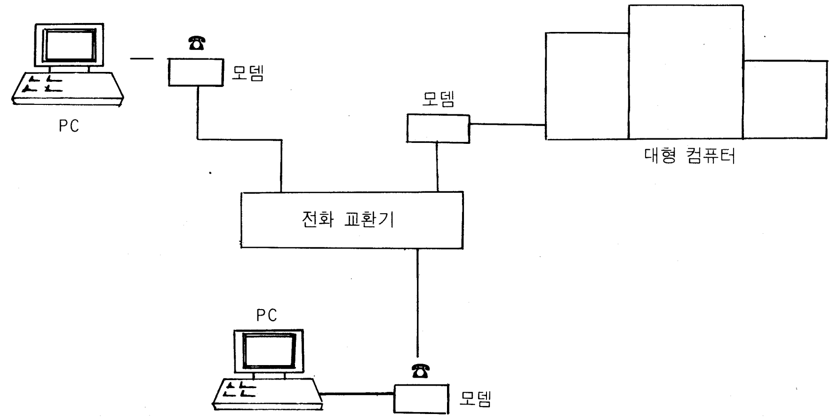
전화기는 사람의 음성을 전기신호로 바꾸어 전화선로를 통해 전달할 수 있도록 해준다. 컴퓨터 신호는 전기신호이므로 꼭 바꾸지 않아도 전화선로를 통해 전달이 가능하나 사람의 음성신호와 신호의 크기나 주파수 등이 다르고 중간에 전화교환기를 거치면서 신호에 이상이 생길 수 있다. 따라서 컴퓨터가 사용하는 전화기와 같이 전화선로에 적합하도록 컴퓨터 신호를 바꾸어 주는 역할을 하는 모뎀이 필요하다(그림 3 참조).

또 사람이 전화를 할 때 전화벨이 울리고 "여보세요"로 시작하는 통화규칙이 있는 것처럼 컴퓨터 사이에도 통신을 하는 규약이 있다. 이것을 "프로토콜"이라 부르는데 모뎀은 이런 통신 프로토콜도 담당한다. 모뎀의 통신규약은 CCITT(국제 전신전화 자문 위원회)나 전화기의 발명 원조인 Bell회사의 규격을 따르는 것이 보통이다.

●화일서버(File Server)

통신을 위해서는 상대방이 통신을 준비하고 있어야 한다. 하지만 미리 컴퓨터 네트워크를 연결하자고 일일이 말로 통화를 하고 통신을 시작하는 것은 상당히 불편하다. 이때 컴퓨터와 컴퓨터사이에 중계소를 만들어 놓고 송신을 원하는 컴퓨터가 자료를 그 중계소에 보내고 수신측 컴퓨터가 중계소에 연결될 때 자신에게 온 것이 있으면 받아가는 방법이 편리하다. 이와 같은 역할을 하는 장치를 화일서버라 부른다. 앞서 말한 전자사서함이나 분담작업도 화일서버를 이용한다.

〔그림 3〕 공중 전화선로 교환망



실제 컴퓨터 네트워크에서 화일서버의 역할은 아주 중요하다. 네트워크에 연결되는 여러 대의 컴퓨터 중에서 식별번호에 의해 사용자를 구별하고 또 사용자의 등급에 따라 이용할 수 있는 자료의 범위를 결정하는 등 네트워크의 주인처럼 움직여야 한다.

(3) 근거리 통신망(LAN)

LAN(Local Area Network)은 통신을 필요로 하는 다수의 컴퓨터들이 가까운 거리에 있을 때 이들을 통신망으로 연결하는 것이다. 통신은 먼거리에만 필요한 것이 아니라 가까운 거리에 있는 사람들이 서로 밀접한 관계에서 일을 하다 보면 빈번한 대화나 통신을 필요로 한다. 그리고 한사무실 안에 여러대의 컴퓨터가 있다면 하나의 디스크 안에 담겨 있는 자료를 공동으로 이용하거나 (화일서버) 컴퓨터마다 각각 프린터를 갖고 있는 것보다 근거리통신망(LAN)을 구성해서 같이 사용하는 편이 공간도 절약되고 효율적이다.

LAN은 다음과 같은 특징이 있다.

1. 비교적 한정된 범위에서 구성된다 (몇미터에서 수킬로미터이내)
2. 통신선로가 한회사의 소유이다.(회사내의 사설전화기나 공청TV 신호선, 또는 특별히 가설된 선로 등으로 법이나 공중규약에 제한을 받지 않는 선로)
3. 통신망 내에 기술적으로 교환을 해주는 기능이 있다.(Switching Technology)

(4) 게이트웨이와 WAN

하나의 LAN에서 다른 네트웨크로 연결하려면 특별한 창구를 통해야 한다. 이 창구를 게이트웨이(Gateway)라 한다. LAN보다 넓은 거리의 컴퓨터들이 망으로 연결될 때는 WAN(Wide Area Network)이 된다. WAN은 거리가 멀기 때문에 개인이나 한회사가 선로를 소유하지 못하고 보통 공중통신망(공중전화선)을 이용하는 것이 보통이다.

(5) ISDN(종합정보통신망)

선진국에서는 고도 정보화 사회로 가기 위해 국가의 중추신경으로서 ISDN을 구축하려고 하고 있다. ISDN은 디지탈 전화망을 발전시킨 통신망이며, 이 망은 여러 가지의 통신기기를 디지탈 접속형태로 연결하여 다양한 서비스를 제공한다. 예컨대 TV, 전화선, 컴퓨터 통신회선 등을 광섬유와 같은 하나의 통신선로에 공동으로 연결하여 이용하는 것이다. 즉 모든 정보가 고속으로 전달될 수 있는 종합적인 정보의

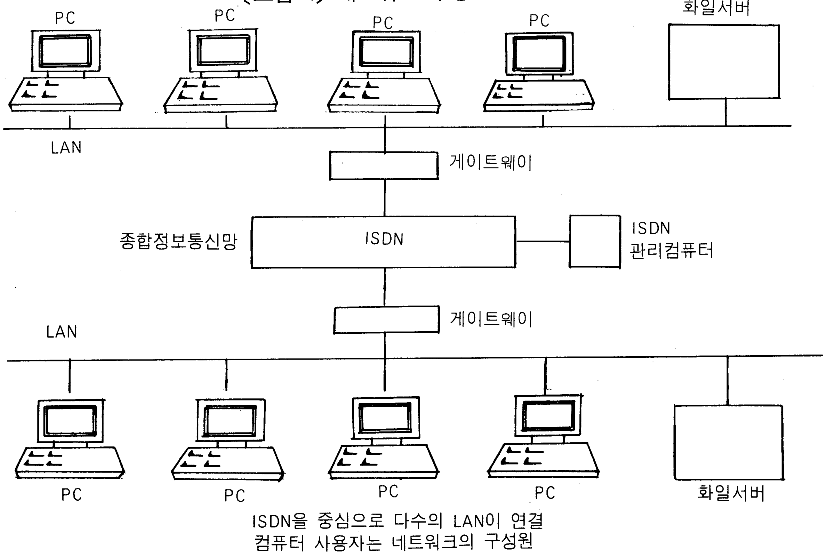
고속도로 통신망이다(그림 4 참조).

(6) 컴퓨터와 통신개념의 변화

컴퓨터와 컴퓨터 사이를 연결하는 통신이 시작될 때는 컴퓨터를 중심으로 해서 외부의 네트워크를 생각했으나 LAN 및 ISDN(종합정보 통신망) 등 통신망이 발전하자 오히려 네트워크를 중심으로 가까운 주변의 컴퓨터들이 연결된 종합적인 시스템으로 이해하는 편이 쉬워졌다. 즉 내가 갖고 있는 컴퓨터의 자료만을 처리하는 것이 아니라 통신망에 연결된 모든 컴퓨터의 자료와 정보를 이용하는 시스템으로 변하고 있다. 그러므로 통신망에 연결된 컴퓨터는 네트워크의 한 가족 구성원이 되는 것이다.

컴퓨터와 컴퓨터가 결합하여 완전한 한 가족이 되기 위해서는 아직 많은 장애가 있지만 결국은 사람과 사람 사이를 연결하는 진보된 컴퓨터 기술이 등장할 것이다. PC

〔그림 4〕 네트워크 구성



〈참고문헌〉

1. 마이크로 컴퓨터 입문, 이기현, 한국공업표준협회, 1982
2. 컴퓨터의 맥, 김려성, 주식회사 민컴 출판부, 1986
3. 근거리 통신망, 정진욱, 변옥환, 옴사, 1986
4. 컴퓨터 네트워크의 시대, 남택선랑(일), 통신정책연구소역, 1987

〈용어해설〉

● **인덱스 홀(Index hole)**
섹터의 개시위치를 나타내기 위해 플로피디스크에 뚫린 구멍.

● **트랙(Track)**
자기드럼, 자기디스크, 자기테이프 등의 표면에 일련의 데이터를 기억시키고, 하나의 헤드로 리드 또는 라이트할 수 있는 동심원.

● **네트워크(Network)**
데이터통신 시스템에서 컴퓨터 및 단말기를 접속하기 위해 쓰이는 기기 및 선로 등으로 구성되는 전송매체로 전송기기, 교환기 등을 포함하는 전송매체가 해당된다.

● **데이터 전송(Data transmission)**
원격지점 사이에 데이터를 송수신하는 것을 말하며, 일반적으로는 데이터 통신시스템에 대한 데이터 전달수단을 말한다.

● **CAI용 언어**
학습자가 컴퓨터와 대화방식으로 학습하는 교육시스템을 쉽게 활용하기 위한 전용 프로그래밍 언어.

● **통신 프로토콜(Communication protocol)**
수신장치가 송신된 데이터를 올바르게 해석하기 위해 직렬 데이터의 흐름에 동기의 숫자나 캐릭터를 사용하여야 한다는 것을 지정하고 있는 규정.

● **RS-232C**
EIA(Electronic Industries Association)가 정한 데이터 단말장치와 데이터 통신장치와의 인터페이스에 관한 규정으로 20k보오(baud) 이하의 속도로 15m 이내의 직렬 데이터의 전송을 할 수 있다.

● **직렬전송(Serial transmission)**
데이터를 시간적인 계열로 하나의 통신선로로 순차 전송하는 것. 즉 1번에 1비트를 전송하는 것으로 원격통신에 있어서 하나의 전신 또는 데이터 신호를 구성하는 신호 엘리먼트를 연속적인 시간간격을 가지고 보내는 것.

● **병렬전송(Parallel transmission)**
다수의 통신회로를 이용하여 동시에 전송을 하므로 전송시간이 짧고 매우 효율적이다.

● **플로피디스크 장치(Floppy disk unit)**
폴리에스테르 판위에 자성 재료를 칠하여 염화비닐 자킷에 넣는 기록매체를 플로피디스크라 하며, 이 플로피디스크를 일정한 속도로 회전시켜 자기헤드로 데이터를 기록하거나 판독하는 장치.

● **섹터(Sector)**
1개의 트랙을 몇개로 분할한 블록으로 레코드라고도 불린다. 보통 리드와 라이트는 이 섹터의 단위로 되고 1트랙당 16~20개로 등분되어 있지만 유저가 섹터의 길이를 프로그램으로 지정할 수 있다.

국내소식

-과학기술처
중소기업 MIS 개발 추진

과학기술처는 중소기업업무 전산화 추진을 돕기위해 경영활동 전반을 컴퓨터로 처리할 수 있는 경영정보시스템을 개발하여 보급할 방침이다. 전산화 사업의 효율적인 추진을 위해 제조업중에 기계, 자동차부품, 전기, PCB(인쇄회로기판), 전자, 화학 및 화공, 섬유, 신발 등을 시범업종으로 선정하여 이들 업종의 경영정보화에 필요한 표준소프트웨어를 우선적으로 개발키로 했다.

이에 따라 1차로 컴퓨터기종과 작업지침서 등 시스템개발 운영환경 등을 표준화하고 각 경영분야의 표준소프트웨어를 통합하여 경영정보시스템으로 발전시키기로 했다.

경영정보시스템은 사무관리분야에서 영업, 인사, 급여, 회계, 재무, 원가, 자금, 무역 등과 조업관리분야에는 자재, 창고, 구매, 생산, 품질, 설비 등이 포함되어 있다.

과기처는 경영정보시스템이 개발되면 5개 시범업종별로 유망업체를 선정하여 보급한뒤 연차적으로 전국에 보급을 확산시키기로 했다.

-상공부
국산컴퓨터 리스제도 강화

상공부는 국산컴퓨터의 보급확대와 수입대체를 추진하기 위해 현행 국민투자기금으로 지원되는 국산컴퓨터 리스제도의 제출서류 및 절차를 대폭간소화하는 한편 신용카드 판매방안 도입 및 컴퓨터리스판매 전문회사 설립을 적극 검토하는 등 컴퓨터리스제도를 강화하기로 했다.

이는 컴퓨터의 내수확대와 함께 컴퓨터리스의 수요가 매년 30%이상 증가하고 있는데 비해 현행 국민투자기금으로 지원되는 국산컴퓨터 리스제도가 과다한 제출서류와 복잡한 절차로 리스사 및 컴퓨터수요자 모두가 이용을 꺼리는 등 많은 문제점을 지니고 있어 국산컴퓨터 리스제도의 제출서류 및 절차를 대폭 간소화한 것으로 알려졌다.

상공부는 이를 위해 현재 전자공업진흥회에서 확인해주는 국산컴퓨터 확인 절차를 없애고, 리스사가 컴퓨터제조회사 확인만으로 자체 판단할 수 있도록 인정기준을 변경하여 한국은행 등 관계금융 기관에 통보했다.

또한 컴퓨터리스관련 서류정비를 위해 올해중 전자진흥회, 리스사, 제조회사 등의 개선위원회를 설치하여 소액컴퓨터리스의 특례안을 마련하여 이를 시행 및 운용할 방침이다.

한편 상공부는 현재 국산컴퓨터 리스자금이 국민투자기금중 국산시설 대여자금으로 되어 있어 컴퓨터가 아닌 소액의 타 산업용기기 리스자금으로 지원되는 점을 감안하여 국산컴퓨터 리스지원자금을 별도 항목으로 배정하고 이의 지원을 강화키 위해 한국은행과 협의중에 있다.

-상호신용금고연합회
공동전산망 구축작업 추진

전국상호신용금고연합회는 최근 연합회와 신용금고, 신용금고간의 데이터교환을 위해 공동전산망 구축을 위한 컴퓨터 및 관련 소프트웨어를 도입키로 하고 제품 제안설명회를 개최하는 등 구체적인 실무작업에 들어 갔다.

이와 함께 동 연합회는 현재 전산화가 완료된 170여개 신용금고 컴퓨터와 네트워크 형성이 가능하며, 전산화 추진 신용금고의 퍼스컴과 데이터를 쉽게 교환할 수 있는 시스템을 올해중 설치, 완료할 방침이다.

또한 동 연합회는 공동전산망 구축을 통하여 금고관리업무 등 연합회업무는 물론 세금우대저축의 중복통장관리, 금융기관의 불량거래자 정보관리와 금고간 거래체제 구축 등 전국의 상호신용금고 업무를 완전 전산화할 계획이다.

-체신부
정보통신서비스 기술개발계획 마련

체신부는 오는 90년대 고도정보화시대에 대비하여 관련정보통신 서비스기술을 본격 개발키로 하였는데 구체적인 계획은 다음과 같다.

우선 올해부터 정보변환장치와 비ISDN(종합정보통신망) 단말기를 ISDN 망에 연결시키는 단말기정합장치 등을 단계적으로 개발할 계획이다.

한편 전자통신연구소와 관련업계가 주측이 되어 텔렉스에 워드프로세서 기능을 추가한 텔리텍스와 텔렉스간의 접속서비스를 제공하기 위한 정보변환 장치를 개발할 방침이다.

체신부는 오는 91년 상용화를 목표로 비ISDN단말기를 ISDN단말기에 연결시키는 단말정합장치를 개발하여 퍼스널컴퓨터를 전화망에 접속하여 각종 통신서비스를 제공할 수 있는 PC통신망 서비스기반을 조성할 계획이다.

이와 함께 업계와 공동으로 텔리라이팅의 국내 표준기술개발 및 상대방의 얼굴을 보며 통화할 수 있는 비디오폰을 비롯한 단말기 개발도 함께 추진키로 했다.

-한국정보산업연합회
PC통신 세미나 개최

한국정보산업연합회(회장 : 이용태)는 지난 3월24일 전경련회관 3층 대회의실에서 500여명의 관계자들이 모인 가운데 'PC통신과 PC FAX 개발활용'이라는 주제로 세미나를 개최했다.

이번 세미나는 (주)삼성전자 후원으로 성균관대 정보공학과 정진욱교수의 PC통신의 개발동향과 운용전망에 대한 강연과 삼성전자 주변기기 개발실 이상화과장의 PC FAX 활용대책 강연으로 진행되었으며 정보의 축적과 가공이 가능한 특징을 갖는 PC통신이 새로운 형태의 통신서비스로 선호될 전망이라는 것과 PC를 팩시밀리 대용으로 여러 용도에 사용할 수 있다는 내용으로 참석자들의 관심을 끌었다.

-한국과학기술원
전문인력의 수급방안 제시

한국과학기술원은 과학기술정책연구평가센터의 21세기 과학기술인력 장기수요 전망에 따르면 2001년까지 필요로하는 박사급 인력은 5만3천명, 석사급 인력은 11만7천5백명으로 추산되자 고도 정보화 사회로 가기 위해서는 과학기술인력의 양적확대보다 산업계에서 필요로 하는 질적개선이 시급함을 지적했다.

특히 전문인력 양성이 시급한 것으로 지적되어 전문인력의 효과적인 수급을 위해 정부에서 운영하는 공공훈련기관을 만들어 노년층, 실업상태 과학기술자, 비과학기술인력 등을 재교육시켜 활용하는 방안을 제시했다.

이 방안은 교육기관을 통해 여성인력은 소프트웨어 개발과 전산작업 등에 노년층과 과학기술인력은 각종 연구개발계획 결과의 평가에 활용한다는 것이며, 이밖에 대학의 기초과학 연구체제를 확립하여 대학별 특성을 살려 연구소를 만들고 국가에서 재정지원을 해주는 방안도 제시했다.

-한국정보산업연합회
각급기관 전산교육 문제점 지적

한국정보산업연합회가 국내 236개 전산화기업 및 각급기관을 대상으로 실시한 전산요원 교육실태 조사에 따르면 국내 전산화 기관들은 지난해중 전산요원들에 대한 전산교육을 한 회사당 연평균 43.6회에 걸쳐 2673시간 정도 실시했는데 전체 응답기관의 60.3%가 교육성과에 대해 회의적인 반응을 나타냈다.

이에 따라 전산실기 및 업무능력 부족으로 인한 별도의 재교육이 교육투자 및 시간의 낭비요소로 작용하고 있는 것으로 지적됐다.

특히 전산교육이 전산부서장의 판단 등 기타 비계획적(68.9%)으로 실시되고 있어 기존업무수행의 장해 및 시스템 운용상 차질을 초래하고 있는 것으로 밝혀졌다.

또한 전산전문 교육기관에서의 교육강사의 책임감 결여와 자질도 문제점으로 지적되고 있다.

사외 위탁교육기관들이 현업에 적용할 수 있는 다양한 교육프로그램을 갖추지않아 동일한 내용을 반복교육하는 등 피교육자의 교육욕구에 부흥치 못한 것으로 분석됐다.

정보산업연합회는 이같은 문제점을 해결하고 전산요원교육의 활성화를 위해 우선 각사가 사내에서 현업을 중심으로한 OJT(교육과 집단연수) 및 사외교육 등을 실시하고 교육계획 등을 현업업무에 직접 활용할 수 있도록 수립해야 한다고 지적했다.

-금성반도체
휴대용 컴퓨터 한글화 성공

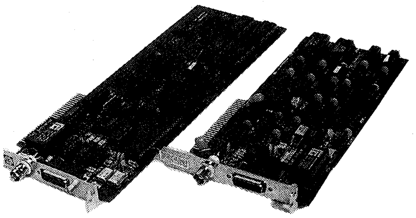
금성반도체(대표 : 이만용)는 상품의 제조일자, 생산지, 일련번호 등 상품의 고유정보를 선부호로 표시한 바코드(BAR CODE)를 스캐너를 통해 판독하여 이를 데이터로 저장하는 휴대용 컴퓨터(PDT : Portable Data Terminal)를 한글화하는데 성공했다.

이로써 현장에서 제품을 취급하는 현장관리자는 누구든지 사용할 수 있어 유통 및 공장자동화에 크게 기여할 것으로 기대된다.

이번에 한글화된 휴대용컴퓨터는 기존 바코드 시스템에 접속하여 데이터를 처리하는 퍼스컴(PC)의 기능을 대신할 수 있는 장비로 신발, 의류 등의 매장판매 현황파악 및 재고관리, 병원의 약품관리, 각종 제조업체에서의 창고관리 등에 폭넓게 사용할 수 있다.

이 휴대용컴퓨터의 특징은 본체에 프린터를 부착하여 저장된 자료의 출력과 간단한 영수증처리가 가능하며 한글의 경우 가로8자, 세로4행을 디스플레이할 수 있어 이미 기록된 데이터와 현재 기록될 데이터를 비교하기가 용이하고 전원이 끊어져도 수은전지가 내장되어 있어 데이터를 안전하게 보존할 수 있다.

-금성반도체
근거리통신망 개발



금성반도체(대표 : 이만용)가 국내 처음으로 근거리통신망(LAN)을 개발했다.

이번에 개발된 인텔리전트 스타랜(Star Lan)은 기존 스타랜이 IBM PC만으로 망의 구성이 가능했던 점을 보완하여 IBM PC기종외의 다른 기종의 단말기로도 스타랜을 구성할 수 있어 그 활용폭이 넓다.

또한 이터네트(Ether Net)의 일종인 치퍼네트는 LAN보드에 중앙처리장치와 송수신기(Tranciver)를 내장시켜 데이터 전송속도와 정보처리 능력이 향상되었다.

그외에 저가격의 이더네트(Low Cost EtherNet)도 개발되었는데 3종류의 LAN모두가 MS-DOS와 UNIX-OS에서 운용가능하고 운용되는 소프트웨어는 국제 표준기구(ISO) 표준안을 채택, 다른 공중통신방식과 접속이 용이하며 NET-BIOS 인터페이스를 제공하여 응용프로그램 개발이 쉽게 되어 있다.

금성반도체는 이번에 개발된 LAN의 판매와 더불어 기존 PC유저들을 위한 LAN보드 및 응용소프트웨어의 개발판매도 병행할 방침이다.

-금성산전
국산 스카다시스템 개발 성공

금성산전(대표 : 이희종)은 전력설비, 수처리설비, 생산관제설비 등 대규모 플랜트의 계통관리를 종합적이고 효율적으로 수행할 수 있는 대형 SCADA(집중원방감시제어) 시스템을 국내 처음으로 개발하였다.

이번에 성공한 동 시스템의 개발로 한국전력을 비롯하여 도시가스 등 대규모 산업설비의 계통관리 자동화에 필요한 대형 SCADA 시스템의 수입을 완전히 대체할 수 있을뿐 아니라 국내실정에 맞는 최적시스템을 공급할 수 있게 되었다.

금성산전 연구소가 개발한 동 시스템은 32비트 마이크로프로세서를 내장하여 감시 제어성능을 크게 강화했으며, 산업설비 자동화의 필수요소인 이중화 구조를 채택하여 메인시스템의 고장시에도 자동 또는 수동스위치를 통해 연속운영이 가능하도록 하여 시스템에 대한 신뢰성을 크게 강화시킨 것이 특징이다.

또한 현장의 응용상황에 따라 시스템소프트웨어 변경없이 관리요원이 작성한 자동제어 프로그램만으로 연속수행을 가능케 하여 시스템의 적용범위가 대폭 확장됐다.

금성산전은 지난 78년 미 해리스사와 기술제휴로 이 사업에 참여한 이래 TADCOM-2000 등 중·소형 스카다 시스템 국산화에 성공하여 시판에 나서고 있으며, 홍콩 및 인도네시아 등의 동남아시아 수출에도 적극 나설 계획인 것으로 알려졌다.

-다우기술
2바이트 조합형 디베이스Ⅲ+ 발표

다우기술은 작년말 KS 2바이트 완성형 코드를 개발한데 이어 데이터관리용 소프트웨어인 2바이트 조합형 디베이스 Ⅲ+를 발표했다.

이로 인해 국내에서 판매되고 있는 대부분의 퍼스컴에서 한글, 한자 디베이스 Ⅲ+ 소프트웨어를 활용할 수 있게 되었다.

이번에 발표된 제품은 여러 유형의 데이터를 상호결합하여 분류할 수 있는 관계형 데이터베이스로 응용프로그램의 작성과 수정이 용이하며 네트워크기능을 보유한 것이 특징이다.

-삼보컴퓨터
CeBIT '89 전시회 참가

삼보컴퓨터(대표 : 김종길)는 3월8일부터 15일까지 서독 하노버(Hannover)에서 개최되는 CeBIT '89전시회에 단독 부스를 마련하고 TG-386V/286V의 신개발품을 비롯하여 모뎀, Music Creating System, Voice Mailing System 등 첨단 관련제품을 선보였다.

85년부터 매년 동 전시회에 참가하여 유럽각국의 자사 브랜드제품의 수출향상에 큰 성과를 올리고 있는 삼보컴퓨터는 서독에 있는 지사를 현지판매법인으로 승격시키고, 약 55만불을 투자하여 자사브랜드의 유럽진출를 꾀하고 있는 것으로 알려졌다.

한편 Hannover Messe는 미국 라스베가스(Las Vegas)에서 매년 개최되는 COMDEX /FALL에 비해 참가업체수가 2,731업체로 1.8배, 방문객은 485,800명으로 5.1배('88년 기준)로 세계 최대규모의 컴퓨터, 컴퓨터 통신 및 주변기기 전시회이다.

-삼보컴퓨터
경기지역에 행망용 시스템설치 완료

삼보컴퓨터(대표 : 김종길)는 금번 국가적으로 시행하는 행정전산망 작업에 참여하여 지난 4월까지 경기도 전지역 읍면 동사무소에 트라이젬 시스템의 설치를 완료했다.

이번에 설치된 시스템은 AT 404대, XT 908대, 프린터 908대로 금년 12월까지 시범운용된 후 90년 1월부터는 주민관리업무에 활용될 예정이다.

한편 현재 경기지역 각 읍·면·동 사무소에서 관리하고 있는 세대별 주민등록표 및 개인별 주민등록표에 관련된 데이터는 약2,000에서 5,000건으

로 이를 제닉스 운영체계와 286시스템을 호스트로 이용한 멀티유저 시스템으로 운영하게 된다.

–삼성전자
랩탑컴퓨터 대량 수출



차세대 컴퓨터로 각광받고 있는 휴대용의 랩탑 컴퓨터가 해외시장에 대량 수출된다.

관련업계에 따르면 87년도에 랩탑컴퓨터를 개발, 88년부터 수출해온 삼성전자(대표 : 강진구)는 PC의 본고장인 미국을 비롯하여 영국, 독일, 프랑스 등 유럽지역과 호주, 동남아시아, 아프리카 등에 금년중 3만대 이상을 수출할 것으로 알려지고 있다.

삼성이 수출하고 있는 랩탑은 가스 프라즈마(Gas Plasma)방식의 S5200과 LCD(Liquid Crystal Dsiplay : 액정 디스플레이) 방식의 S5000 등 2가지 모델이다.

휴대용 컴퓨터의 세계시장 규모는 86년에 88만대 20억불을 기록했으며, 89년에는 50%이상 성장한 140만대 32억불에 이를 전망으로 일본의 NEC와 도시바, 미국의 IBM 등 세계굴지의 컴퓨터회사 제품이 주종을 이루어온 랩탑 시장에 삼성전자가 본격 참여함으로써 시장에 삼성전자가 본격 참여함으로써 국내컴퓨터 산업의 기술력을 세계시장에서 평가받게 되었다.

한편 삼성전자는 금년 하반기부터는 한글 / 한자 소프트웨어를 내장한 랩탑 컴퓨터를 국내시장에 선보일 예정이다.

–삼성전관
City Card-10K 도입

삼성전관은 키보드에 직접붙여서 사용할 수 있는 카드리더인 City Card-10K를 경기시스템(주)에서 OEM방식으로 공급받기로 계약하고 1차분 120대, 2차분 30대를 도입하였다.

삼성그룹내 Topics(종합정보서비스)에 대한 보안유지 및 카드 미소지자의 사용 불가능, 키보드를 이용한 간편한 조작과 운영을 목적으로 도입된 City Card-10K는 현재 NEC-5200 터미널에 부착되어 사용되고 있다. 삼성은 그룹내에 이 시스템을 계속 확대보급시킬 방침이다.

한편 증권회사에서도 증가하는 증권투자가들에게 증권카드에 대한 서비스 확대를 위한 이 시스템의 도입을 검토하고 있는 것으로 알려지고 있다.

–삼성전관
광학식 문자판독기사업 진출

삼성전관(대표 : 김정배)은 최근 일본 NEC사가 개발한 소형 OCR인 N6-370UM을 도입하여 시판에 나섰다.

금융, 보험, 증권업을 비롯하여 유통서비스업, 의료 및 교육기관 등의 각종 전표처리작업에 적합하도록 설계된 이 OCR은 전표를 분당 최대 50매까지 처리할 수 있을 뿐만 아니라 일련번호별로의 자동인쇄가 가능해 1대로 전문 전표처리인력(키펀처) 15명분의 일을 수행한다. 이밖에 자체내 오류방지기능을 지니고 있어 어떠한 호스트컴퓨터에서도 접속해 사용할 수 있다.

–선경
퍼스컴 해외 수출 계약

종합무역상사 선경은 대만의 퍼스컴 생산업체인 마이텍과 전격적으로 국내 독점 디스트리뷰터 계약을 맺은데 이어 최근에는 유럽, 미주, 중동지역에도 자사 브랜드를 부착하여 수출하는 사업에 합의했다.

이에 따라 선경은 호환 퍼스컴인 마이텍의 XT, AT, 386의 올해 수출목표를 5만대로 잡고 지속적인 고객지원과 판매망 확보를 위해 현지 CE요원을 채용하면서 서독의 함부르크지사, 미국의 뉴욕지사, 중동의 리야드지사를 통해 판매거점을 마련하는 등 본격적인 수출을 위한 모든 준비를 서두르고 있다.

반면에 국내공급은 국내 독점딜러인 한국마이크로텍을 통해 마이텍브랜드로 판매될 방침이다.

–쌍용컴퓨터
쌍용양회의 FA 프로젝트 추진

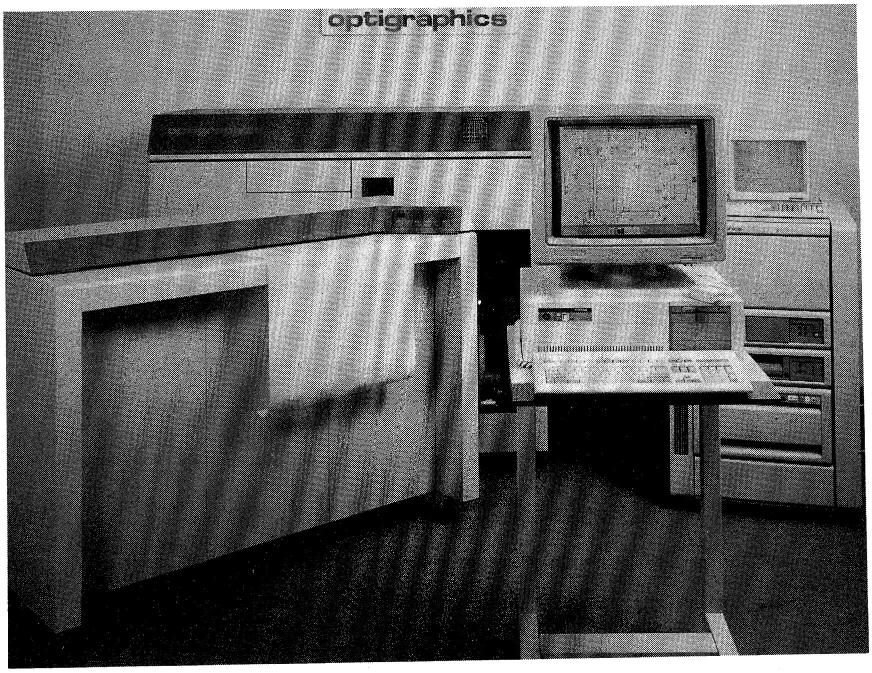
쌍용컴퓨터(대표 : 최병환)는 최근 그룹 계열사인 쌍용양회 영월공장의 대규모 2단계 공장자동화 프로젝트를 수주했다.

쌍용컴퓨터는 이 프로젝트의 추진을 위해 시멘트제조 공정관리상 필요한 각종 정보를 제공하는 "Data Logging System"의 국산화에 성공했으며, 이외에도 중앙제어실에서 제조공정을 상황에 따라 손쉽게 조작할 수 있는 인터페이스용 대규모 소프트웨어도 국산화할 예정이다.

현재 국내 시멘트제조업체의 대부분의 생산설비가 에너지 대량소비형임을 감안할 때 쌍용양회는 이번 공장자동화의 추진으로 획기적인 에너지 절감효과를 볼 수 있게 되었으며, 또한 중앙제어실에서 각 생산공정을 자동관리하여 안정조업을 물론 생산성과 품질향상을 동시에 이룰 수 있을 것으로 전망된다.

–쌍용컴퓨터
도면관리 및 CAD 변환시스템 시판

쌍용컴퓨터(대표 : 최병환)는 최근



건축, 전기, 전자, 기계, 조선분야의 설계도면과 같은 각종 도면정보의 관리 및 제작, 수정, 보관을 효율적으로 처리할 수 있는 도면자료 자동관리 및 CAD(컴퓨터 이용설계) 변환 시스템의 시판에 들어갔다.

쌍용컴퓨터가 미국 옵티그래픽스사와 공급계약을 맺고 국내에 시판하게 된 이 시스템은 종이나 필름 등에 기록된 도면을 판독하여 입력하는 스캐너와 입력된 도면정보를 관리, 검색, 수정하는 소프트웨어 및 편집완료된 자료를 각종 CAD 시스템에 사용할 수 있도록 변환시켜 주는 트랜스레이터 등으로 구성되어 있다.

쌍용컴퓨터가 국내에 시판키로한 제품모델은 IBM-PC와 386제품을 워크스테이션으로 사용할 수 있는 "Optigraphics 1000"과 SUN워크스테이션을 기본으로한 "Optigraphics 3100" 등이다.

이들 제품은 도면의 인식, 수정 등 관리능력면에서 일반 시중제품보다 매핑, CAD 등과의 인터페이스 기능이 우수하고 자동처리 부분과 사람이 조작하는 부분이 잘 조화되어 사용이 간편하다.

이들 시스템의 주요시장은 도면의 제작, 수정, 보관 등을 필요로 하는 엔지니어링회사와 건설회사 및 정유, 제철, 자동차, 조선 등의 대규모 생산설비의 보유회사이며, 이외에도 다량의 도면을 보유, 관리하고 있는 정부기관과 한국전력공사, 한국전기통신공사 등과 같은 정부투자기관이 주요 수요처가 될 것으로 보인다.

-아남반도체
ASIC 설계사업 강화

아남반도체설계(대표 : 윤정우)는 통신기기용 ASIC설계주문이 크게 증가함에 따라 설계능력향상을 위한 이 분야의 사업을 대폭 강화하고 있다.

동사는 지난해 ASIC설계가 9건에 그쳤으나 올해들어 퍼스컴, 주변기기 및 스위칭, 모뎀, 전화기 등 통신기기용 ASIC설계주문이 크게 늘자 설계인원보강뿐만 아니라 회로선폭을 0.85μ까지 향상시켰다.

이는 올해초 VLSI테크놀로지가 일본의 히다찌와 제휴함으로써 히다찌의 0.85μ급 프로세서기술을 확보하게 되었으며, VLSI테크놀로지와 기술제휴계약을 체결한 아남반도체설계도 히다찌의 프로세서기술을 사용할 수 있게 되었기 때문이다.

동사는 현재 20만 게이트까지의 설계능력을 확보하여 게이트어레이, 스탠더드셀, 실리콘컴파일러 등을 설계하고 있다.

한편 동사는 앞으로도 ASIC설계 수요가 계속 증가할 것으로 보고 있으며 올해 20건이상, 내년에는 30~40건을 설계할 계획이다.

-왕컴퓨터 코리아
Wang 음성처리용 통합 어플리케이션 개발 툴 발표

왕컴퓨터 코리아는 음성처리와 전화기능을 통합하는 새로운 소프트웨어 툴인 STEP(Speech and Telephony Environment for Programmers)을 발표했다.

STEP은 Wang VS시스템에서 사용하는 소프트웨어 패키지로 사설교환기(PBX)와 Wang의 WBX(Wang Business Exchange), AT&T의 Centrex 1AESS스위치를 통합시키는 역할을 하며, VS 시스템 프로그래머가 C, PL/1, COBOL언어로 음성 어플리케이션을 쉽게 개발할 수 있는 툴과 환경을 제공한다. 이 패키지를 사용하면 음성처리에 데이터, 화상, 문서 및 통신기능을 통합한 소프트웨어 환경을 구성할 수 있다.

PBX와 연결되어 메시지 대기신호, 전화번호 감지에 의한 송·수신자 규명 등의 기능을 수행하고, STEP으로 개발된 VS 어플리케이션으로 전화를 송·수신하고 조절할 수 있다. 또한 통합화상정보 시스템인 WIIS, VS의 관계형 데이터베이스 관리 시스템인 PACE와 통합사용도 가능하다.

한편 STEP은 왕 VS시스템의 음성처리 주변장치인 VFEP-1000(Voice Front End Processor)를 지원하는데,

이 VFEP는 자체에 프로그램이 가능한 시그널처리 하드웨어가 있어 새로운 음성처리 기술을 쉽게 채택할 수 있다. 하나의 VFEP는 4~16개의 전화선을 지원하며 4개선 단위로 증설할 수 있다. 한 VS시스템에 여러개의 VFEP를 연결이 가능하고 STEP에 의해서 100개까지의 전화선을 수용한다.

-유한시스템

PCB 자동검사장치 개발

유한시스템(대표 : 장을노)은 아주대와 공동으로 컴퓨터를 이용한 인쇄회로기판(PCB) 자동검사장치를 개발하는데 성공하였다.

상공부의 공업기반기술 개발과제로 개발이 추진된 이 제품은 최대 측정핀수가 2천48개로 외국제품보다 많으며 측정속도와 정밀도에 있어서도 우수한 것으로 평가되고 있다.

이 제품은 PCB 단락여부와 동작기능상태의 검사 등 불량여부를 컴퓨터로 측정하는 장치로 16비트 퍼스컴을 주축으로 PCB 측정장치, 스캐너 검사 프레스, 보조기억장치, 디스플레이, 키보드, 프린터 등의 하드웨어와 이들을 서로 연결시켜주는 접속프로그램과 운용프로그램 등으로 구성되어 있다.

또한 이 제품은 데이터의 수정과 측정모델변경이 용이하고, 자체고장진단 프로그램을 내장하여 자동으로 작동의 이상여부를 파악할 수 있는 것이 특징이다.

-큐닉스

오토캐드 국내 디스트리뷰터로 결정돼

오토캐드(Auto CAD)의 국내 디스트리뷰터로 소프트웨어 전문업체인 (주)큐닉스데이터시스템이 결정됐다.

큐닉스데이터시스템(대표 : 김경래)은 지난 4월초 오토캐드개발업체인 미 오토데스크사와 국내 디스트리뷰터계약을 체결하고 오토캐드 국내보급에 나섰다.

이로써 현재 삼성물산을 비롯하여 금성소프트웨어, 캐드랜드, 태일시스템, 다산시스템 등 5개 딜러 업체들의 오토캐드 물량을 큐닉스데이터 시스템이 총괄하게 되었다.

큐닉스데이터시스템은 이번 계약을 통하여 오토캐드 국내 공급물량에 대해 오토데스크사와의 창구를 일원화하고 실사용자들에 대한 교육 및 기술지원 서비스를 높이는 동시에 큐닉스의 한글 DOS기술과 연계하여 오토캐드 한글화 작업을 추진할 방침이다.

-한국IBM

초대형 컴퓨터 발표

한국IBM(대표 : 피터.T.로울리)은 대형 시스템아키텍처인 ESA/370과 운영체제로 MVS/ESA를 채택한 초대형 컴퓨터 ES/3090 모델 100S와 ES/4381 모델그룹 90E를 발표했다.

이번에 발표된 모델은 IBM3090 및 4381모델 가운데 최하위 모델로 가격대 성능비 비율이 높으며, 시스템 유저들의 업무량 증가에 따른 시스템 확장에 적합한 시스템이다.

또한 연산장치가 복수의 운영체제 환경하에서 운영되도록 하는 PR/SM(Processor Resoure/System Manager) 기능이 채택되어 하나의 연산장치를 7가지의 서로 다른 환경에서 사용할 수 있으며, VM/XASP 릴리즈2를 이용하여 보조기억장치에 내장된 자료를 신속하게 입출력할 수 있다.

-엘렉스컴퓨터

매킨토시IIx 발표회 가져

주식회사 엘렉스컴퓨터(대표 : 이윤기)는 지난 4월14일 신제품 발표회를 가졌다. 이번에 개최한 제품 발표회에서는 지난 1월에 미국에서 성황리에 발표한 Macintosh Ⅱx를 비롯하여 Ⅱcx와 CD-SC 등이 선보였다.

이를 제품은 포스트 스크립트, 엔지니어링 워크스테이션, 그래픽 프로세싱, 전자출판, OA용 시스템 등의 용도로 널리 활용이 가능하다.

Mac Ⅱx 및 Mac Ⅱcx는 모토롤러사의 68030 마이크로프로세서를 탑재하고 있는데, 이는 하드디스크의 일부를 RAM으로 사용할 수 있는 가상메모리기법과 시스템 에러가 발생했을 때 실행중이었던 어플리케이션만 재시동하는 메모리 보호기법을 비롯하여 공유 메모리기법 등을 가지고 있다. 또한, 속도가 15.66MHz인 CPU를 채택하여 기존의 유저를 보호하는 의미로 Mac 제품의 완전한 호환성을 유지함을 우선으로 하였다.

Mac Ⅱcx와 Mac Ⅱx의 주메모리는 기본으로 2MB와 4MB를 갖고 있으며 보드상에서는 공히 8MB까지 확장할 수 있을뿐만 아니라 슬롯을 사용하여 64~128MB까지 확장 가능하다.

-갑일전자(주)

황희선 신임사장 취임



(주)갑을은 테레비디오 컴퓨터를 인수하고 지난 4월10일 상호변경과 신임사장취임식을 거행하였다.

회사명은 테레비디오 컴퓨터에서 갑일전자(주)로 변경하였으며 신임사장에 황희선 금성통신 고문이 취임하였다.

갑일전자는 앞으로 책임경영, 최고의 품질향상, 과감한 기술개발투자와 기술인력의 양성, 연구개발체제 확립, 주변기기 부품 등의 신규사업에 주력할 예정이다.

국외소식

마이크로소프트사
경제통계와 중소기업자문을 위한 CD 공급

미국의 마이크로소프트사는 최근에 CD-ROM으로 된 경제통계 팩과 중소기업자문을 위한 소프트웨어를 판매하기 시작했는데, 이것은 빠른 시장동향과 기업의 자문을 필요로 하는 경영자에게 큰 도움이 되리라 생각된다.

통계 팩은 놀랍게도 1987년도 미국의 통계축출, 공유지통계, 금융조사, 각 지역의 임금조사, 농업통계의 다섯가지 연방정부의 자료를 싣고 있다.

이런 데이터들은 스프레드시트 프로그램으로 .WK1과 .WLS로 쉽게 불러서 사용할 수 있다.

이 문서들은 CD(Compact Disc)의 일부분을 차지하고 있으며, 그밖에도 국제통화기금(IMF)의 국제금융통계가 실려 있다.

또 하나의 CD는 중소기업자문을 위한 소프트웨어 수백개로 이루어져 있는데 구체적으로 알아보면 다음과 같다.

즉, SBA(Small Business Administration) 과 행정부에 의해 소규모 사업의 시작, 운영, 확장하려는 수많은 자료를 포함하고 있다. 또한 국제수지를 위해 회계업무를 전담으로 하고 있는 Deloitte Haskins & Sells사로부터 분석을 의뢰할 수 있으며, 이를 통해 받을 수 있는 정보는 사업계획, 재정, 판매 뿐만 아니라 손실 등도 예측하여 미연에 방지할 수 있다.

반도체업계
인텔, NEC 분쟁을 계기로 새로운 돌파구 찾아

지난 2월7일, 그때까지 4년간 계속하여 온 NEC-Intel 양사의 마이크로프로세서(마이크로코드) 저작권해에 대한 분쟁이 일단 막을 내리고 "종지부"를 찍었다. 이 결말은 당사자 2사에 주어진 승패보다도 결론으로서 내놓은 판결이 반도체, 컴퓨터 양업계에 미칠 광범위할 영향이 훨씬 중대하다.

이 소송문제에 내려진 판결요지는 다음의 3가지이다. ① Intel의 마이크로프로세서에 꾸며 넣어진 마이크로코드는 미국저작권법의 보허대상으로 되는 저작물이라 인정된다. ② Intel은 그 저작물보호를 위해서 최저한의 필요표시장치를 부착하는 것을 소홀히 하였기 때문에 이 저작권의 유효성을 상실하였다. ③ NEC의 마이크로프로세서 V시리즈의 마이크로코드는 Intel의 보호되어야 할 마이크로코드저작권을 침해하고 있지 않다.

이 판결의 결과 앞으로 마이크로프로세서시장(이 시장은 반도체, 컴퓨터 양산업의 요체이다)에 2개의 큰 영향을 미친다고 생각된다. 하나는 클론 마이크로프로세서의 "자유화"이며, 또 하나는 범용(CISC)에서 전용(RISC)으로 마이크로프로세서의 주류이전에로 더욱 박차가 가해지는 것이다.

우선 상기 판결취지 3가지(특히 ①과 ③)를 종합하여 볼 때 다음과 같은 점이 앞으로 문제가 되리라고 생각된다.

「NEC의 V시리즈가 8086, 8088의 마이크로코드를 침해하고 있지 않다」고 하는 것은「호한 마이크로프로세서를 만든다고 하는 목적실현을 위해서 일정한 제약조건은 있어도 그 수단이 합법적인 것이면—즉 마이크로코드라는 저작권을 침해하지 않으면—, 그 결과(기능)의 유사성은 위법은 아니다」고 하는 호환 CPU개발에 전혀 새로운 규범을 제기한 것으로 된다.

이 사고방식은 IBM PC 호환기로 널리 채용되고 있는 BIOS의「크린 룸(격리)개발 / 리버스 엔지니어링」을 정당화하여 인지하는 것으로 된다.

또한 16비트 CPU시대까지는 상식적으로 전개되어 온「세컨드 소스」라는 IC의 라이센스생산·판매형태도 필연적으로 그 의의가 엷어지게 된다. Intel이 32비트 마이크로프로세서(80-386) 시장 과점화전략으로서 세컨드 소스의 실시를 완고하게 거절하여 온 바탕은 NEC의 마이크로코드저작권 불침해라는 판단에서 흔들기기 시작한 것이다. 즉 세컨드 소스의 라이센스가 공여되지 않아도, 그리고 기술적인 곤

란은 있다고 하여도 386, 486의 클론화는 앞으로 합법적으로 실행할 수 있는 것이다. 사실 미국의 신흥 IC메이커의 일부가 1990년 초에는 합법적인 호환 386 CPU를 탑재한 컴퓨터를 시장에 발매할 계획을 진행하고 있다.

캘리포이아주 산호세소재의 Nexgen Microsystems사는 이러한 신흥메이커의 1사에서 Intel386을 사용하지 않고 동사 독자의 486보다 50% 고속인 CPU 세트에서 386 소프트웨어를 실행하게 하는 컴퓨터를 내년에 발표할 계획이다.

그렇다면 마이크로코드판결로 실질적으로 패소하게 된 마이크로프로세서 시장의 패자 Intel은 앞으로 어떤 시장 전략(반격)으로 나올 것인가? 이에 대한 해답으로서 2가지를 생각해 볼 수 있다. 하나는 386의 부분적인 라이센스를 케이스 바이 케이스로 공여하여 갈 것. 또 하나는 클론 CPU메이커와의 차별화대책이다. 후자의 경우 합법/위법을 불문하고 독자의 클론 CPU 개발에 긴 시간과 다액인 자금을 요한다고 하는 호환 IC메이커의 약점을 찾아 「적이 따라 붙기 전에 선행하여 이긴다」는 전략의 전개이다.

사실 인텔의 CPU 개발차별화 전략은 매우 급속도로 추진되고 있다. 지난 2월 중반 뉴욕에서 개최된 ISSCC(국제고체회로회의)에서 M-10의 호칭으로 발표한 64비트의 RISC 마이크로프로세서를 i860으로서 2월27일에 발표하였을 뿐이나 IBM, AT&T, Olivetti, Prime Computer 등에서 이의 채용을 결정하였다(표 1 참조). 또한 개발에 4년의 세월과 3억달러를 들이고, 또 호환대책으로서 휨웨어화를 시행한 32비트 CPU 80486도 금년 상반기중에는 출하되리라 생각된다.

이번 판결의 결과 뜻밖에 Intel과 마찬가지로 NEC도 또 V시리즈라는 CISC 범용CPU 노선에 박차를 가하여 지난 2월초순, MIPS Computer System사의 RISC CPU의 세컨드 소스로 확정

〔표 1〕 RISC 마이크로프로세서 공급 상황

메이커	제품명	세컨드 소스	주요 사용 컴퓨터 메이커
Sun Micro Systems	SPARC	• 후지쯔 마이크로 일렉트로닉스 • Cypress Semiconductor • BIT • LSI Logic • TI • Prisma	• AT&T • Xerox • Unisys • ICL • Solbourne (마쯔시다)
MIPS Computer Systems	R2000A R3000	• NEC • 시멘스 • LSI Logic • IDT • Performance Semiconductor	• DEC • 소니
Intel	i860 (N-10)		• IBM • AT&T • Olivetti • Prime Computer
Motorola	88100		• Motorola
AMD	2700		

되었다. 양사(NEC와 Intel)가 고속처리시장, 즉 워크스테이션 CPU시장에 동시에 활발한 움직임을 보이는 것은 참으로 흥미있는 일이다.

NEC-Intel의 16비트 마이크로프로세서 저작권계쟁은 이미 결말을 보았다. 그러나 이를 계기로 세계 PC시장의 흐름은 확실히 PC(퍼스컴)에서 WS(워크스테이션)로 향하고 있는 듯 하다. 지금이야 말로 32비트 RISC CPU 시장에서 세컨드 소스전개가 활발히 진행되고 있는 것이다. 세대교체가 정착한 1990년대에는 CPU메이커간에서 16비트시대의 전철을 밟는 저작권 분쟁이 일어나지 않으리라는 것을 기대하는 것은 무리가 아닐까 싶다.

영국, 실리콘 글렌지방

유럽의 실리콘 벨리로 자리잡아

1992년 유럽시장의 단일화를 앞두고, 반도체 및 전자분야의 세계적인 거대기업들이 영국의 스코틀랜드에 소재한 실리콘 글렌에 속속 진출함으로써, 이 지역이 유럽의 새로운 전자 공업단지로 부각되고 있다.

세계 최대의 실리콘 웨이퍼 제조업체인 일본의 신에쯔 반도체가 지난 4월 10일 이 지역에 공장을 준공함으로써 실리콘 글렌은 IBM, Digital, Honeywell, Hewlett Packard 등의 미국 회사들과 함께, 연간 1,500억 달러의 유럽시장을 놓고 본격적인 시장 확보전 돌입을 위한 전초기지화 하고 있다.

이와 같이 미·일의 전자기업들이 다투어 이 지역에 진출하고 있는 이유는, 인구 3억 2천의 유럽이야말로, 오는 1992년 세계최대의 단일 시장으로 그 엄청난 수요와 마케팅 활동의 용이성 때문이기도 하지만, 특히 이 지역이 전자공업 특유의 숙련된 노동력을 쉽게 구할 수 있는데다, 외국기업 진출에

따른 영국 정부차원의 각종 혜택이 뒤따르기 때문이라고 한다.

실리콘 클렌은 이미 30년전부터 진출해 있는 IBM과 NCR을 비롯해, Hewlett Packard, 미쓰비시 전자, NEC 등 300개가 넘은 세계적인 반도체 및 전자기업들이 밀집해 있고, 총 43,000여 명의 고급 인력들이 업체에서 활동하고 있으며, 생산된 제품의 3분의 2는 유럽 대륙의 각국으로 팔려나가고 있다.

현재 실리콘 글렌을 포함한 스코틀랜드 지역에서는 유럽 반도체 시장의 20%가 생산되고 있으며, 1992년까지는 유럽 내 마이크로 컴퓨터의 40% 가량이 이곳에서 생산될 전망이다.

이번에 공장을 준공한 일본의 신에쯔 반도체 역시 공장 준공과 동시에, 수백만달러가 드는 제2의 초정밀 반도체 생산공장 건립계획을 발표했으며, 이 회사의 유다로 고사케사장은 "이번에 준공된 공장에는 총 1,400만불을 투입했으나, 신에쯔 반도체는 향후 수년간에 걸쳐 5,400만불을 실리콘 글렌에 투자할 것"이라고 함으로써, 이 지역에 대한 미래를 예견해 주었다.

실리콘 글렌을 미국의 실리콘 벨리에 맞설만한 규모로 육성하기 위해, 영국 정부의 노력 또한 대단하여, 이 지역 개발을 위한 별도의 스코틀랜드 산업 개발청을 설치, 각국 투자기업들에 대한 특별 지원을 실시하고 있으며, 작년 한햇 동안의 이 지역에 대한 외국 기업들의 신규 투자만도 무려 7억 5천만불에 달함으로써, 실리콘 글렌은 명실공히 유럽의 실리콘 벨리로 자리잡아 가고 있다('실리콘 글렌'의 ' 글렌'은 스크틀랜드 지방어로 '밸리'란 뜻이다).

IBM사
하이퍼카드와 같은 기능의 「Linkway」 발매

샌프란시스코에서 개최된 MacWorld EXPO에서 Silicon Beach Software사는 HyperCard 호환 S/W인 「SuperCard」를 발표하여 큰 호응을 받았었다. 이러한 HyperCard툴은 앞으로의 멀티미디어 · 어플리케이션 소프트웨어 개발에 큰 역할을 담당할 것은 명백한 사실이다. 그러나 IBM PC 시장에서는 멀티미디어·소프트웨어라고 하면 OWL사의 「Guide」가 있을 뿐이며 실제의 보급면에서 보면 매킨토시에 비해서 크게 뒤져있는 실정이다.

따라서 IBM사가 이러한 점을 감안하여 판매하기 시작한 것이 「Linkway」라고 불리우는 HyperCard라이크인 멀티미디어 어플리케이션 개발툴이다. 이 툴은 작년에 발표한 것이지만 그 자세한 내용에 대해서는 지금까지 명확하게 알려져 있지 않았다. 그러나 이번에 정식으로 판매를 개시함과 동시에 개발세미나를 개최하여 멀티미디어 개발자에게 상세한 것을 소개하였다.

Linkway를 사용하면 IBM PC나 그 호환기종에서 텍스트, 컬러그래픽스, 풀모션비디오, 음악, 음성 등을 자유로 조합한 복잡한 어플리케이션을 개발하는 것이 가능하게 된다. Linkway를 사용해본 관계자는 「Linkway로 만든 대부분의 어플리케이션은 1행의 프로그래밍도 필요로 하지 않는다. 어플리케이션개발자는 마우스를 사용하여 메뉴를 선택하여, 옵젝트를 연결하는 것만으로 하이퍼액세스가 가능한 프로그램을 구축할 수 있다」고 그 조작의 편리함을 강조하고 있다.

Linkway의 가격은 싱글유저판이 110달러, LAN대응판이 660달러 정도이며, 매킨토시의 HyperCard를 의식하였는지 상당히 저가격으로 되어 있다. IBM사의 교육부문에서 Linkway를 담당하는 Tony Peacock은 Linkway가 ① 컬러를 서포트하고 있다. ② 염가인 DOS상에서 동작한다(최소 구성은 384Kbytes RAM+FDD 1대+마우스), ③ Linkway의 페이지기술방식은 비트맵방식은 아니고 폰트맵방식이다 등의 특징을 가지고 있는 것을 들어 HyperCard 의 직접적인 카피제품은 아닌 것을 강조하고 있다.

호주, 로보트론사
신체장애자용
「유리카 A4 전자비서기」 개발

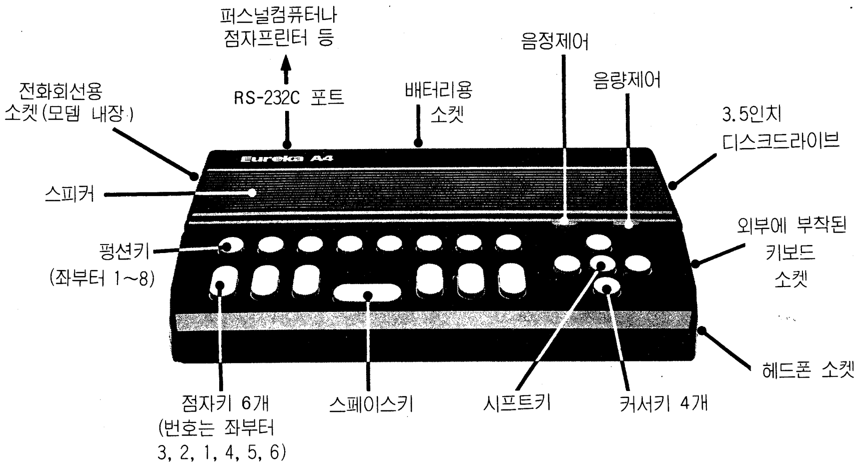
호주의 로보트론사에 의해 개발된 「유리카 A4 전자비서기」는 가수인 스티브 원더도 사용하고 있을 정도로 「비서기」라는 이름이 적합하도록 여러가지의 다양한 기능을 가진 신체장애자용 컴퓨터이다. 표 2에서 보는 바와 같이 여기에는 8개의 펑션키가 있는데 이것을 그냥 누를 경우와 쉬프트키와 동시에 누를 경우 등 전부 16가지의 기능이 있다.

유리카A4 전자비서기는 작곡기능이나 통신기능, 온도계, 전압계 등도 있

〔표 2〕 유리카 A4의 기능

f·1	노트기능
f·2	알람시계/카렌더
f·3	과학시계기능을 가진 범용 전탁
f·4	통신기능
f·5	전화장/오토다이얼 기능
f·6	컴퓨터 기능 (BASIC 인터프리터)
f·7	작곡기능
f·8	디렉토리 표시
시프트+f·1	와이프로 기능
시프트+f·2	일기/메모장
시프트+f·3	온도계
시프트+f·4	전압계
시프트+f·5	데이터베이스 기능
시프트+f·6	디스크 유틸리티
시프트+f·7	디스크에서 프로그램 실행
시프트+f·8	디스크 코맨트

(사진 1) 유리카 A4의 외형



으며, 단지 비즈니스용 뿐만 아니라 취미로도 사용할 수 있도록 되어 있는 것이 특징이다.

워드프로세서나 데이터베이스 등의 입력용에는 6개의 점자키를 사용하며 통상의 키보드(IBM PC / XT용)를 따로 접속하여 사용하는 것도 가능하다(사진 1 참조). 입력그레이드의 설정에 의해서 "bit"라 입력하려면 "b"만, "about" 라 입력하려면 "ab"만을 타이프(세계공통)하는 등 타이핑을 단축하는 기능도 있다.

디스플레이는 부속하고 있지 않으며 모두 음성으로 확인하도록 되어 있다. 어떤 문자를 입력한 것인가, 지금 어떤 기능이 있는 것인가, 그리고 현재의 배터리상태 등 여러가지의 체크가 가능하며 데이터의 보존용으로 3.5인치(포맷시의 용량은 792Kbytes)의 플로피디스크 드라이브를 내장하고 있다. 여기서 보존한 데이터는 유리카전용이지만 IBM 기종에도 텍스트데이터를 읽을 수 있도록 전용의 변환 소프트웨어가 첨부되어 있다.

미, Solbourne사
SPARC CPU를 사용하는 워크스테이션 개발

NeXT사의 NeXT Computer System, DEC사의 DEC station 3100(MIPS Computer 사의 RISC칩을 탑재한 14-MIPS를 자랑하는 최신머신) 등의 등장으로 활발한 경쟁을 하고 있는 워크스테이션업계에 새로이 Sun Microsystem사의 SPARC CPU를 사용하는 워크스테이션이 등장하였다.

미국, 콜로라도주에 본거지를 가지는 Solbourne사는 Sun-4시리즈와 같은 SPARC CPU를 사용한 워크스테이션을 개발하였다고 발표하였다(덧붙여 말하면 Solbourne사는 송하전기의 후원을 받고 Sun클론을 개발하여 왔다). 이번에 발표한 것은 「Solbourne 시리즈4」라고 불리우는 기종. 최고 4개의 CPU를 멀티로 처리할 수 있도록 설계되어 있지만 CPU 1개의 구성으로는 Sun-4와 마찬가지로 약10MIPS의 연산속도를 가진다. SunOS, SunView 및 NeWS윈도우, NFS 등 Sun-4와 똑같은 환경을 제공하고 있으며 소프트웨어는 Sun사에서 OEM공급받는다.

가격은 Sun-4의 동위기종보다 낮게 설정되어 있으며, 16Mbytes RAM, 327Mbytes HDD, 150Mbytes 테이프 드라이브의 구성에서 약 5만달러 정도이다. 또한 동사는 1년 이내에 ULS판의 SPARC를 탑재한 고속모델을 발표할 계획이다.

카스웰컴퓨터사
음성출력기 「퍼스널 리더」발표

미국 카스웰 컴퓨터사가 개발한 「퍼스널 리더」는 영문의 책이나 신문 등을 스캐너로 읽어 들여 대단히 명료한 음성으로 읽어 내는 컴퓨터이다.

스캐너는 인텔리전트 OCR(광학문자판독기)이라고 불리우는 시스템이며 문자의 서체나 크기, 하이픈 등이 섞여 있는 문장에서도 높은 인식률로 판독한다. 그리고 판독한 문장은 ASCII코드로 변환되어 본체에 기억되므로 퍼스널컴퓨터에 전송하여 데이터를 보존하기도 하고, 편집하는 것도 가능하고, 퍼스널컴퓨터에서의 데이터를 음성화할 수도 있다.

음성화는 본체에 내장된 DECtalk보드가 하고 있다. 이것에 의해서 음질은 남성의 음성이나 여성의 음성 등 9종류 중에서 선택되기도 하고, 읽어내는 속도의 조절(매분120어~350어)도 가능하다.

콘트롤이나 읽어낼 때의 조작 등은 키보드를 사용하여 한다. 물론 각 키의 조작설명도 음성으로 들을 수 있는 것이 특징이다.

음성에 의존하지 않고 또 점역 등을 하지 않고 문자를 직접 읽는 것으로 손가락의 촉각을 이용하는 「옵타콘」이 있다. 문장을 손쉽게 음성으로 변환하여 주는 퍼스널 리더는 유감스럽지만 영어로 밖에 대응하고 있지 않지만(장래 불어 등 5개국어에 대응시킬 예정) 장시간의 이용이나 대량의 문서를 읽을 경우를 생각하면 획기적인 장치라고 할 수 있을 것이다.

디렉토리의 결합과 분리

종종 응용 프로그램을 작성하다 보면 현재의 디렉토리내에서 다른 디렉토리를 연결하여 사용할 필요성이 있다. 도스버전 3.3에서는 이러한 디렉토리의 결합과 분리기능을 제공하고 있는데 간단한 예를 통하여 이러한 기능을 알아보고 응용해 보자.

어떤 프로그램을 실행시키고자 할 때 프로그램과 데이터가 서로 다른 드라이브의 서브디렉토리에 저장되어 있을 경우 프로그램 상태에서 데이터를 이용하려면 그 데이터가 저장된 디렉토리를 찾아가야 한다. 이럴 경우 데이터가 들어 있는 드라이브를 같은 서비디렉토리로 취급해 주는 기능이 있다면 프로그램과 데이터가 같은 드라이브상에 존재하는 것과 같아 사용상의 편리함을 제공할 수 있다.

이와 같이 한 드라이브를 다른 드라이브의 디렉토리에 논리적으로 연결하여 단일 디렉토리구조로 만들려면 JOIN 명령어를 사용하면 된다. 드라이브 A에 **화면 1**과 같은 내용이 저장되어 있을 때 드라이브 A를 패스 C:\DOOLY에 연결시키려면 다음과 같이 한다.

```
C>JOIN A: C:\DOOLY
```

라고 하면 드라이브 A가 C 드라이브의 서브디렉토리 DOOLY에 연결되어 마치 C 드라이브의 서브디렉토리와 같이 취급된다. DIR명령으로 **화면 2**와 같이 A 드라이브의 전체가 DOOLY에 연결되었음을 확인할 수 있다.(실제로 읽어들일 때는 A 드라이브에 읽어들인다는 것을 알 수 있는데 이는 물리적으로 연결된 것이 아니라 논리적으로 연결되어 있기 때문이다) 이것은 지정된 드라이브의 현재 디렉토리와 관계없이 그 드라이브의 전체 트리가 연결되기 때문이다.

여기서 지정된 디렉토리가 패스에 존재하지 않으면 지정한 드라이브에서 새로이 디렉토리가 만들어 진다. 그리고 그 디렉토리가 존재하면 그것은 반드시 비어 있어야 한다. 빈 디렉토리에는 시스템화일인 MSDOS.SYS와 IO.SYS만이 감춰진 화일(Hidden file)로 들어 있다. 예를 들어 C:\DOOLY의 디렉토리가 비어 있을 때 "dir C:\DOOLY"를 타입하면 다음과 같이 디스플레이된다.

```
Volume in drive C has no label
Directory of  C:\DOOLY

.            <DIR>      1-01-80  12:16a
..           <DIR>      1-01-80  12:16a
        2 File(s)   249856 bytes free
```

만일 JOIN하고자 하는 디렉토리가 비어 있지 않을 경우에는 "Directory not Empty"라는 메시지가 디스플레이된다. 또한 연결된 드라이브는 참조할 수 없다. 만일 참조하려면 "Invalid drive specification"이라는 에러 메시지가 나타난다.

〔화면 1〕

```
A:\>dir

 Volume in drive A has no label
 Directory of  A:\

TAL            1408   1-01-80  12:06a
JUBO           1536   1-01-80   5:38a
JOIN           2560   1-01-80  12:14a
PE             1152   3-28-89  12:07a
HJ1            2048   3-28-89   1:23a
D         <DIR>       1-01-80  12:16a
        6 File(s)   329728 bytes free
```

〔화면 2〕

```
C:\DOOLY>dir

 Volume in drive C has no label
 Directory of  C:\DOOLY

TAL            1408   1-01-80  12:06a
JUBO           1536   1-01-80   5:38a
JOIN           2560   1-01-80  12:14a
PE             1152   3-28-89  12:07a
HJ1            2048   3-28-89   1:23a
D         <DIR>       1-01-80  12:16a
        6 File(s)    47104 bytes free
```

이것은 디폴트 드라이브가 유효하지 않기 때문이다.

이와 관련되어 기억할 것은 두 디렉토리가 연결되어 있는 동안 BACKUP, RESTORE, FORMAT, DISKCOPY, DISKCOMP 명령들을 사용할 수 없다.

그리고 현재의 디렉토리를 패스로 지정할 수 없으며 지정하는 디렉토리 패스는 루트 디렉토리가 될 수 없다.

현재의 디렉토리가 어떻게 연결되었는지 알고 싶다면

```
C>JOIN
```

라고 하면 다음과 같이 연결시킨 드라이브와 연결된 디렉토리 패스와의 관계를 나타내는 표시가 디스플레이된다.

```
A: => C:\DOOLY
```

앞에서 설명했던 것처럼 A드라이브는 JOIN되는 동안은 참조할 수 없다. 그러므로 A 드라이브를 사용하려면 연결기능을 해제하여 원래의 상태로 회복시켜야한다. 그럼 JOIN되었던 것을 풀려면 어떻게 해야 할까? 이럴 때는 다음과 같이 하면 된다.

```
C>JOIN A: /D
```

라고 하면 JOIN이 제거되었으므로 이후부터는 드라이브 A를 참조할 수 있다. 끝으로 기억할 것은 JOIN 명령어는 외부 명령어이므로 실행시킬 때는 JOIN.EXE 화일이 제공된 상태에서 해야하며, 같은 드라이브는 JOIN할 수 없다. PC

전자분야 10년의 역사를 한눈에 확인할 수 있습니다.



월간 전자과학에서는
1979년 1월호부터 1988년 12월호까지
총 120권을 장정본으로 묶어 한정판으로 판매합니다.
서둘러 구입하십시요.
전자분야에 관한 종합적이고
체계적인 자료집으로서 지난 10년간의 변천과정과
앞으로의 미래를 예측할 수 있게 해드립니다.

□ 구입방법

본 장정본의 구입을 원하시는 분은 시중 은행에서
지로용지를 이용하시거나 우체국에서 대체구좌 또는
소액환으로 대금을 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256-31-0526772

□ 구입가격

20만원

□ 문의처

월간 전자과학 영업부　서울시 용산구 갈월동 17
TEL : (02)754-1044　FAX : (02)757-1049

월간 **전자과학** 월간 **퍼스널컴퓨터** 도서출판 **오므사** 관인 **한국전자통신학원**

첨단서적

컴퓨터 전문서적 통신판매 써비스

도서명	출판사	가격	도서명	출판사	가격	도서명	출판사	가격
표준 MS DOS 핸드북	기전연구사	7,000원	알기 쉬운 dBASE Ⅲ	영진	6,000원	SIDEKICK plus	성안당	6,000원
MS DOS 핸드북	정보시대	3,000원	dBASE Ⅲ plus 프로그램 모음집	영진	9,000원	GW BASIC 3.2	성안당	4,500원
MS DOS 입문	성안당	5,000원	dBASE Ⅲ plus	크라운	7,000원	GW BASIC	크라운	5,500원
MS DOS	성안당	5,000원	dBASE Ⅲ plus	기전연구사	6,000원	IBM BASIC	성안당	6,000원
IBM DOS 3.2	크라운	5,000원	Auto CAD 2.6	영진	15,000원	워드 퍼펙트	우진출판사	7,000원
MS DOS 3.3	크라운	6,000원	Auto CAD 실무	크라운	5,500원	PC TOOLS DELUXE	크라운	3,000원
알기 쉬운 MS DOS	영진	5,000원	Auto CAD 실습과 응용	영진	10,000원	터보파스칼 4.0	크라운	8,000원
고급과정 MS DOS	크라운	6,000원	C언어 기초+α	교학사	8,000원	WORDSTAR 4.0	우진	6,000원
MS DOS 핸드북	크라운	9,000원	Run C	교학사	7,000원	WORDSTAR 실무	영진	4,500원
DOS 3.3	영진	8,000원	C언어의 이론 응용	성안당	7,000원	애플 Ⅱ 입문	영진	3,500원
MS DOS 4.0	연학사	6,500원	C 프로그래밍	동일	5,000원	Symphony 메뉴얼	크라운	7,000원
MS DOS 메뉴얼	크라운	4,000원	C언어 500제	크라운	5,000원	Symphony 메뉴얼	영진	7,000원
MS DOS와 GW BASIC	연학사	5,500원	C언어 입문과 실습	크라운	5,500원	RM COBOL	성안당	5,000원
Lotus 123 기본	크라운	5,000원	C 라이브러리	영진	5,000원	MS COBOL	성안당	6,000원
실무 Lotus	크라운	6,000원	TURBO C(User's)	크라운	5,000원	IBM PC 어셈블리	기전연구사	6,000원
Lotus Leal 2	크라운	5,000원	TURBO C(Reference)	크라운	5,000원	IBM PC 하드웨어 종합분석	성안당	5,500원
완벽 Lotus	영진	5,000원	MS C 교과서	성안당	6,000원	MSX DOS 입문	연학사	5,500원
알기 쉬운 Lotus	크라운	9,000원	Clipper Compiler	영진	5,000원	MSX DOS 슈퍼 핸드북	우진	4,500원
Lotus 123	기전연구사	6,000원	실무자를 위한 Crosstalk	영진	5,500원	MSX BASIC 컴퓨터교실	기전연구사	4,000원
실무 Lotus 123	영진	10,000원	노턴 유틸리티	영진	3,500원	컴퓨터 용어사전	영진	4,500원
실무 Lotus 123(제2장)	영진	12,000원	노턴 유틸리티	크라운	4,000원	컴퓨터 용어사전	대은	9,000원
dBASE Ⅱ	크라운	4,000원	XENIX V 메뉴얼	영진	7,000원	미니탭 메뉴얼	크라운	6,000원
dBASE Ⅲ	기전연구사	5,500원	XENIX 오퍼레이팅 시스템	크라운	5,500원	APPLE Ⅱ 종합분석	성안당	5,000원
실무 dBASE Ⅲ plus	영진	6,000원	Dr. HALO Ⅱ	교학사	3,000원	APPLE BASIC 프로그램 실습	영진	6,000원
초보자를 위한 dBASE Ⅲ plus	영진	6,000원	Dr. HALO Ⅲ	교학사	3,000원	인공지능	영진	4,000원
프로그래머를 위한 dBASE Ⅲ plus	영진	6,000원	센서 인터페이싱	기전연구사	24,000원	마이크로프로세서	동일	7,000원

※ 본지에 게재된 기타 신간서적도 주문 가능함

- **주문방법** : 직접 구입 또는 교재명과 출판사명을 기재하여 해당금액에 발송료 500원을 추가하여 우체국 소액환으로 동봉 신청하면 즉시 우송함.
- **보 낼 곳** : ㊥140-012 서울시 용산구 한강로2가 전자단지 17동 3층 가열 325호
  전화 : 716-6024

전문인을 섬기는 **첨 단 서 적**

곱셈과 나눗셈

박승제 /여주컴퓨터학원장

8088은 그 구성에 있어 기본적인 골격은 8080을 이어 받았지만 훨씬 더 융통성있고 다양한 명령체계를 가지고 있다. 그 중의 하나가 바로 곱셈명령과 나눗셈명령이다.

8080은 곱셈명령과 나눗셈명령이 없다. 그러므로 곱셈을 하기 위해서는 곱해지는 수(피승수)를 곱하는 수(승수)만큼 덧셈을 반복하여 처리하며, 나눗셈의 경우에는 반대로 반복적으로 빼는 과정을 반복한다.

그러나 8088에는 곱셈명령과 나눗셈명령이 있기 때문에 복잡한 식이라도 비교적 간단히 처리할 수가 있다. 이번 호에는 곱셈명령과 나눗셈명령, 그리고 분기(jump)명령에 대해 알아 보자.

곱셈명령

[형식] MUL 〈오퍼랜드〉

곱셈명령을 사용하기 위해서는 곱하는 수(승수)가 레지스터의 값이든 메모리의 내용이든 상관이 없지만, 곱해지는 수(피승수)는 반드시 AL레지스터(혹은 AX레지스터)에 넣어두도록 하고 있다. 그러므로 MUL명령은 곱해지는 수를 기술하지 않아도 AL아니면 AX일 것이므로 8088에서는 제1 오퍼랜드를 생략하게 된다.

승수의 크기가 8비트라면 자동적으로 AL레지스터와의 곱셈을 수행하며, 그 결과는 AX레지스터에 저장된다. 또 곱하는 수가 16비트의 크기이면 AX레지스터와의 연산을 하게 되며, 그 계산결과는 DX+AX에 저장된다. 이때 DX에 상위워드, AX에 하위워드가 기억된다.

곱셈은 8비트×8비트=16비트, 16비트×16비트=32비트로 답이 구해지므로 곱하는 수의 크기에 따라 AL 혹은 AX가 자동적으로 선택되는 것이지만 그 크기가 확실하지 않는 경우에는 지난 호에서 배운 PTR 연산자를 이용해서 그 형을 명시하여야만 한다.

```
MUL DATA
........
........
DATA DB 4
```

위의 경우에는 DATA의 형이 바이트(DB)로 지정되어 있으므로 PTR연산자를 생략해도 된다.

```
MUL [BX]
```

이번에는 BX의 실제값이 바이트인지 워드인지 분간하기 어렵게 된다. 그러므로 이때는 PTR 연산자를 기술하여 다음과 같이 표기하여야 한다.

```
MUL BYTE PTR [BX]
```

혹은

```
MUL WORD PTR [BX]
```

또, 곱셈명령에 있어서는 상수와 직접 곱하는 명령은 없으므로 이 경우에는 다른 레지스터 등을 통해 중계하여야 한다. 즉, AX레지스터의 내용을 5배로 하고 싶은 경우에는 다음과 같이 된다.

```
MOV AL, AH
MOV BL, 3
MUL BL
```

위의 연산에 의해 AH레지스터에 상위바이트가, AL레지스터에 하위바이트가 기억된다.

8비트의 연산이든지 16비트의 연산이든지 계산결과는 다른 레지스터(AX 혹은 DX)에도 영향을 미치게 되므로 필요시 해당 레지스터의 값을 미리 다른 곳에 저장해두지 않는다면 그 값을 잃게 되므로 주의하여야 한다.

부호를 고려한 곱셈

곱셈의 경우 MUL의 명령은 부호를 고려하지 않는 경우의 연산에 사용된다. 부호가 붙은 수치들을 곱할 경우에는 IMUL 명령을 이용하여 곱셈을 하여야 한다. IMUL의 명령은 연산을 하는 과정에서 부호를 고려하여 계산결과의 부호가 결정될 뿐 명령의 형식은 MUL과 같다.

부호를 고려하지 않는 연산의 경우와 부호를 고려한 연산의 경우를 각각 필산으로 해보면 다음과 같다.

```
   12H        12H
 ×F0H      ×-10H
 -----     ------
 10E0H      FEE0H
```

(F0H를 음수로 보면 -10H에 해당한다.)

그러면 예제를 통해 곱셈 명령의 사용방법을 익혀보기로 하자.

〈예제 1〉

메모리에 저장된 2개의 1바이트 수치를 MUL과 IMUL로 각각 곱한 결과를 서로 비교하여 보라. 또, 메모리에 저장된 2바이트의 수치들을 MUL과 IMUL로 곱한 결과를 각각 비교하여 보라.

〈처리 방법〉

메모리의 D1, D2에 각각 1바이트의 수치를 넣어 둔다. 또, D3, D4에는 2바이트의 수치를 넣어 둔다. 계산결과는 각각 ANS1~ANS4에 기억된다.

연산 결과는 지난 호에서 처럼 문자열로 치환하여 출력하되, 실제의 값은 DEBUG.COM을 이용해서 확인하도록 한다.

여기서는 30H와 −10H(F0H), 1234H와 −EE0H(F120H)의 값을 연산하도록 한다. 지난 호에서는 데이터 세그먼트를 코드 세그먼트와 일치시켰지만 이번 호에서는 데이터 세그먼트를 따로 설정하여 처리하도록 한다(리스트 1 참조).

리스트 1에서는 MUL명령은 레지스터와 레지스터끼리 연산을 하며, IMUL 명령은 레지스터와 메모리의 내용과 연산을 하고 있다. 반드시 이렇게 하지 않으면 안되는 것은 아니고, 레지스터와 메모리사이의 연산을 하는 보기에 불과하다.

000CH와 0014H를 보면 하나는 'BYTE PTR'을 생략하고 있고, 0014H는 'BYTE PTR'을 기술하고 있다. 'D1', 'D2'는 바이트(DB)로 선언이 되어 있으므로 'BYTE PTR'을 기술해도 되고, 생략해도 된다.

또 연산결과를 저장하는 과정에서 16비트의 값은 MOV ANS1, AX 등과 같이 한 번의 명령으로 처리하지만 연산결과가 DX와 AX에 저장되어 있는 값은 ANS3이 'DD'로 선언되어 있더라도 한번으로 이동하는 것은 불가능하고, 1워드씩 나누어서 저장하게 된다. 이 경우에는 다음과 같이 번지를 간접으로 지정하게 된다.

```
MOV   BX, OFFSET ANS3
MOV   [BX], DX
MOV   [BX+2], AX
```

즉, ANS3의 번지를 BX에 기억시키고, BX가 가리키는 번지에 DX의 값을 저장하게 된다. 또, BX번지+2([BX+2]) 에는 AX의 값을 저장한다. 이와같이 레지스터의 값을 메모리의 주소로 세트시켜 두고, 그 레지스터의 값을 참조하는 방식을 '간접 지정'이라고 한다.

간접 지정방식은 아무 레지스터나 다 쓸 수 있는 것은 아니고 BX, BP, SI, DI의 4종류만 가능하다. 또한 이 레지스터와 숫자를 조합해서 번지를 표현하기도 한다. 이때 조합할 수 있는 경우를 표시하면 다음과 같다.

1) BX, BP, SI, DI 자체의 값
 ex) MOV [BP], AX
 ex) MOV [DI], AX
2) BX / BP+8비트 / 16비트 수치('/'는 선택을 나타냄)
 ex) MOV [BX+3], AX
 ex) MOX [BP+2], AX
3) SI / DI+8비트 / 16비트 수치
 ex) MOV [SI+2], AX
 ex) MOV [DI+2], AX
4) BX / BP+SI / DI
 ex) MOV [BX+SI], AX
 ex) MOV [BX+DI], AX
5) BX / BP+SI / DI+8비트 / 16비트 수치
 ex) MOV [BP+SI+3], AX
 ex) MOV [BP+DI+3], AX

이때 '+3'과 같은 것을 변위(Displacement) 라고 하는데 변위는 [BX+3] 처럼 꺽쇠괄호('[]') 안에 기술할 수도 있고, [BX]+3 처럼 꺽쇠괄호 밖에 기술해도 된다.

위의 조합방식중 실제의 기계어에는 BP만을 사용하는 명령은 없고, BP+n (n은 수치)의 형식만 있다. 그러나 프로그래머가 BP만을 지정하게 되면 자동적으로 BP+0으로 어셈블러가 대치하게 된다.

```
MOV [BP], AX  ——  MOV [BP+0], AX
```

8088에서는 번지를 간접 지정하는 경우에는 그 번지들이 모두 데이터 세그먼트에 있다고 간주하게 된다. 그러므로 프로그램의 선두에는 항상 데이터 세그먼트를 설정하는 부분이 필요하게 된다. 이것은 데이터 세그먼트를 코드 세그먼트와 일치 시키든 별도로 설정하든지간에 필요한 과정이다.

위의 예제를 필산으로 처리하면 다음과 같다. 여기서는 30H와 −10H(F0H), 1234H와 −EE0H(F120H)의 값을 연산하도록 한다.

```
(1)      30H
       × F0H
       ------
       2D00H
(2)      30H
       × −10H
       ------
       FD00H
(3)     1234H
       ×F120H
       ---------
       11253A80H
(4)     1234H
       × −EE0H
       ---------
       FEF13A80H
```

(−10H는 부호를 고려하지 않으면 F0H가 된다.)
(FD00H는 −300H이다.)
(−EE0H는 F120H가 된다.)
(FEF13A80H는 −10EC580H이다.)

위의 프로그램을 실행시키면 화면에 이상한 문자만 찍히므로 실행결과로는 계산의 정확성 여부를 알 수가 없다. 그러므로 DEBUG상에서 실행시켜 확인하여야 한다. 화면 1에 DEBUG에 의한 실행결과를 확인하는 과정을 표시했으니 참고하기 바란다.

나눗셈명령

[형식] DIV 〈오퍼랜드〉

나눗셈명령은 16비트÷8비트=8비

트, 32비트÷16비트=16비트의 몫이 구해지도록 되어 있다. 그러므로 나누어지는 수(피젯수)는 AX 혹은 DX+AX에 저장하여야 한다.

나눗셈명령은 그 형식에 있어 곱셈명령과 동일하다. 즉, 제1 오퍼랜드는 AX 혹은 DX+AX이어야 하므로 제1 오퍼랜드는 생략하게 된다.

오퍼랜드는 곱셈과 마찬가지로 8/16 비트의 범용 레지스터이거나 메모리의 수치이다.

연산의 결과는 AL에 나눗셈의 몫이, AH에 그 나머지가 구해진다. 32비트를 16비트로 나눈 경우라면 AX에 그 몫이, DX에는 나머지가 저장된다. 만약 IMUL에 의해 부호를 고려한 나눗셈의 경우에는 나머지의 부호는 나누어지는 수(피젯수)의 부호를 따르도록 되어 있다.

불능과 오버플로우의 처리

수학에서는 0으로 나누는 경우를 불능이라고 하며, 연산할 수 없다고 규정짓고 있다. 컴퓨터에서는 0으로 나눈 경우에는 'Division by zero'에러가 발생하게 되는데 이것은 8088 기계어에서는 INT 0 인터럽트에 해당한다. 즉, 'INT 0'명령은 'Divide overflow'라는 메시지를 출력한 후 프로그램을 종료하고 DOS로 복귀하는 일을 한다.

나눗셈의 연산시 0으로 나누는 경우에는 프로그래머가 지정하지 않더라도 자동적으로 INT 0가 호출되며, 프로그램의 수행은 그 시점에서 중단되게 된다.

또, 0으로 나누지 않더라도 계산결과 몫이 16/8 비트를 넘어서는 경우에도 이와 같은 에러가 발생한다. 무슨 말이냐 하면 16비트÷8비트=8비트, 32비트÷16비트=16비트 크기의 몫이 구해지는 것으로 하고 있는데 FFFFH÷1의 경우는 FFFFH가 되어 몫이 16비트가 된다. 이것은 위에서 말한 16비트÷8비트=8비트의 규칙에 어긋난다.

〔리스트 1〕 부호를 고려한 곱셈

```
0000                          MULTIPLE    SEGMENT
                                          ASSUME  CS:MULTIPLE, DS:DATA
0000  B8 ---- R                           MOV     AX, DATA
0003  8E D8                               MOV     DS,AX
                                          ;
0005  A0 0000 R                           MOV     AL, D1
0008  8A 1E 0001 R                        MOV     BL, D2
000C  F6 E3                               MUL     BL
000E  A3 0006 R                           MOV     ANS1, AX
                                          ;
0011  A0 0000 R                           MOV     AL, D1
0014  F6 2E 0001 R                        IMUL    BYTE PTR D2
0018  A3 0008 R                           MOV     ANS2, AX
                                          ;
001B  A1 0002 R                           MOV     AX, D3
001E  8B 1E 0004 R                        MOV     BX, D4
0022  F7 E3                               MUL     BX
0024  BB 000A R                           MOV     BX, OFFSET ANS3
0027  89 17                               MOV     [BX], DX
0029  89 47 02                            MOV     [BX+2], AX
                                          ;
002C  A1 0002 R                           MOV     AX, D3
002F  F7 2E 0004 R                        IMUL    WORD PTR D4
0033  BB 000E R                           MOV     BX, OFFSET ANS4
0036  89 17                               MOV     [BX], DX
0038  89 47 02                            MOV     [BX+2], AX
003B  C7 47 04 3024                       MOV     [BX+4], '0$'
                                          ;
0040  B4 09                               MOV     AH, 9
0042  BA 0006 R                           MOV     DX, OFFSET ANS1
0045  CD 21                               INT     21H
                                          ;
0047  B4 4C                               MOV     AH, 4CH
0049  CD 21                               INT     21H
004B                          MULTIPLE    ENDS
                                          ;
0000                          DATA        SEGMENT
0000  30                        D1          DB    30H
0001  F0                        D2          DB    -10H
0002  1234                      D3          DW    1234H
0004  F120                      D4          DW    -0EE0H
0006  0000                      ANS1        DW    0
0008  0000                      ANS2        DW    0
000A  00000000                  ANS3        DD    0
000E  00000000                  ANS4        DD    0
0012                          DATA        ENDS
                                          END
```

이 경우에도 INT 0이 호출된다는 의미이다.

INT 0은 우리가 'DOS 기능호출'로 사용하는 INT 21H 등과 함께 인터럽

〔화면 1〕 예제 1의 실행결과 확인

```
C>DEBUG ———— DEBUG를 기동
-N LIST-1.EXE —— Load할 화일이름을 지정
-L ——————— 지정한 화일을 Load함
-R ————— 레지스터의 내용을 표시          프로그램의 크기
AX=0000  BX=0000  CX=0062  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1056  ES=1056  SS=1066  CS=1066  IP=0000   NV UP EI PL NZ NA PO NC
1066:0000 B86B10        MOV    AX,106B ——— 이 값이 데이터 세그먼트이다.
-D 106B:0,20 —— 데이터 세그먼트의 0~20번지의 내용을 화면에 표시
106B:0000  30 F0 34 12 20 F1 00 00-00 00 00 00 00 00 00 00   0.4. ..........
106B:0010  00 00 24 30 5D C3 55 8B-EC FF 36 CE 5C B8 E2 08   ..$0].U...6.\..
106B:0020  50                 미리 저장해 둔 데이터(상위와 하위가 바뀌어 있다.)
-G=0,62 ———— 처음부터 끝까지 실행
 Program terminated normally
 -R
 AX=0000  BX=0000  CX=0062  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
 DS=1056  ES=1056  SS=1066  CS=1066  IP=0000   NV UP EI PL NZ NA PO NC
 1066:0000 B86B10        MOV    AX,106B
 -
 -D 106B:0,32 ———— 데이터 세그먼트의 내용을 표시
 106B:0000  30 F0 34 12 20 F1 00 2D-00 FD 25 11 80 3A F1 FE   0.4. ..-..%..:.
 106B:0010  80 3A 24 30 5D C3 55 8B-EC FF 36 CE 5C B8 E2 08   .:$0].U...6.\..
 106B:0020  50 E8 D9 52 83 C4 04 80-7E 04 03 75 05 B8 E5 08   P..R....~..u...
 106B:0030  EB 03 B8                                          ...
 -Q                             연산 결과

 C>
```

트 명령이라고 부르는 것으로 인터럽트에 대한 자세한 것은 다음에 배우게 된다. 이런 점들만 유의하면 나눗셈의 명령은 별로 어려울게 없다. 그러면 예제를 가지고 나눗셈의 명령을 실습하기로 한다.

〈예제 2〉

윗변이 100, 밑변이 150, 높이가 75인 사다리꼴의 넓이를 구하라.

〈처리 방법〉

실제 이런 문제를 어셈블리로 해결하려는 사람들은 없을 것이나 여기서는 나눗셈의 예를 실습하는 것으로 이해해주기 바란다. 넓이=(가로+세로)×높이/2이다. 데이터를 메모리에 두는 경우도 있겠으나 여기서는 간단한 수치이므로 레지스터끼리 연산을 한 후에 값을 메모리에 저장하는 것으로 한다(리스트 2 참조).

예제 2의 연산을 필산으로 하면 (100+150) ×75/2이다. 그런데 250에 75를 곱하면 2바이트(16비트)의 수치가 되고, 이것은 2로 나눈 후에도 16비트의 자료가 되므로 16비트÷8비트=8비트 형식으로 나눗셈을 하게 되면 'Devide overflow' 에러가 발생하게 된다. 그러므로 이것은 32비트÷16비트=16비트 형식에 맞추어서 계산한다.

```
0012  F6 E3     MUL    BL
```

연산에 의해 곱셈 결과는 AX에 저장되므로

```
0014  99        CWD
```

에 의해 32비트 자료로 확장을 하게 된다. 연산결과의 몫은 AX에, 나머지는 DX에 구해지는데 위의 연산은 나누어 떨어지므로 나머지는 0이 될 것이다.

답은 메모리에 저장되므로 어셈블 후 DEBUG상에서 수행하여 결과를 확인하여야 한다(화면 2 참조).

〈예제 3〉

2시간 47분 45초와 3시간 57분 52초의 덧셈을 수행하여 그 답을 시, 분, 초로 구하라.

〈처리 방법〉

시간의 계산은 맨 아랫자리(초)부터 덧셈을 한 후 그 윗자리로 자리올림을 해야 한다. 위의 문제를 필산으로 하면 다음과 같다.

```
   2시간  47분  45초
+  3      57    52
   ─────────────────
   5     104    97 ──→ 6시간 45분 37초
                              ↓
                        6H, 2DH, 25H
```

초에서 분으로 자리 올리기를 하는 과정은 다음과 같다.

(1) 초를 60으로 나눈 몫과 나머지를 구한다. 이때 나머지가 '초'이다.

(2) (1)에서 구한 몫을 '분'에 더한다.

이것은 분에서 시간으로 자리올리기를 하는 경우에도 동일하다. 위의 연산은 자리 올리기를 고려하여 연산하면 복잡하므로 먼저 시간, 분, 초를 각각 더하여 메모리에 기억시켜 두고 자리올리기는 한꺼번에 처리하기로 한다(리스트 3 참조).

화면 3은 DEBUG를 이용해서 실행결과를 확인해 본 것이다.

비교 명령

[형식] CMP 〈제1 오퍼랜드〉, 〈제2 오퍼랜드〉

컴퓨터가 탁상용 계산기와 다른 점은 어떤 값의 대소를 스스로 비교 판단할 수 있다는 점이다. 어셈블리에서는 비교하는 명령으로 CMP라는 것

이 있다. CMP는 〈제1 오퍼랜드〉와 〈제2 오퍼랜드〉의 크기를 비교하는 것인데 이때 사용 가능한 오퍼랜드의 조합은 다음과 같다.

(1) 범용 레지스터(8 / 16)와 범용 레지스터(8 / 16)
(2) 범용 레지스터(8 / 16)와 메모리의 내용(8 / 16)
(3) 범용 레지스터(8 / 16)와 수치(8 / 16)
(4) 메모리(8 / 16)와 범용 레지스터(8 / 16)
(5) 메모리(8 / 16)와 메모리(8 / 16)
(6) 메모리(8 / 16)와 수치(8 / 16)

비교의 결과는 '크다', '작다', '같다', '크거나 같다', '작거나 같다', '같지않다' 등으로 표시할 수 있으며 이것은 분기 명령(JMP)과 혼합하여 사용된다.

분기명령

[형식] JMP 〈레이블〉

분기(jump)명령은 무조건 분기와 조건부 분기로 나눌 수 있다. 조건부분기는 위의 CMP명령의 연산결과에 의한 분기와 플래그 레지스터의 내용에 의한 분기로 구별된다. 〈레이블〉은 프로그램상의 위치를 지정하는 의사 명령으로 ':'로 끝나는 일련의 문자열을 말한다.

ex) JMP NEXT ('NEXT:'라는 레이블이 붙은 곳으로 분기한다.)

```
      ...
NEXT:

    JMP NEXT
```

대소비교에 의한 분기명령은 다음과 같은 것이 있는데 비교하는 값이 부호가 붙은 수치인가, 부호가 없는 수치인가에 따라 분기 명령의 기술방법이 다르다(표 1 참조).

표 1을 보면 명령어가 비슷비슷해서

〔리스트 2〕 사다리꼴 넓이 구하기

```
0000                        MAIN    SEGMENT
                                    ASSUME CS:MAIN, DS:DATA
0000  B8 ---- R                     MOV   AX, DATA
0003  8E D8                         MOV   DS, AX
                                    ;
0005  B4 00                         MOV   AH, 0
0007  B0 64                         MOV   AL, 100
0009  B7 96                         MOV   BH, 150
000B  B3 4B                         MOV   BL, 75
000D  B9 0002                       MOV   CX, 2
                                    ;
0010  02 C7                         ADD   AL, BH
0012  F6 E3                         MUL   BL
0014  99                            CWD
0015  F7 F1                         DIV   CX
0017  A3 0000 R                     MOV   ANS, AX
001A  89 16 0002 R                  MOV   ANS+2, DX
001E  B4 4C                         MOV   AH, 4CH
0020  CD 21                         INT   21H
0022                        MAIN    ENDS
                                    ;
0000                        DATA    SEGMENT
0000  0003[                   ANS   DW  3 DUP(0)
            0000
                      ]
0006                        DATA    ENDS
                                    END
```

눈이 돌아갈 지경이지만 명령어의 의미를 기억해두면 혼돈하지는 않는다. 표 1의 의미는 다음과 같다.

J : Jump
G : Greater than(크다)
L : Less than (작다)
E : Equal to (같다)
N : Not (부정)
A : Above (이상)
B : Below (이하)
Z : Zero (0)

위의 명령어중 'N'은 앞에 붙어 부정의 의미를, 'E'는 뒤에 붙어 '같다'는 의미를 부여한다. 즉, '크거나 같다'는 'GE'(Greater than or Equal to)가 된다. 또, '크거나 같다'는 '작지 않다'는

〔표 1〕 부호의 유무에 따른 분기명령

A : B	부호가 있음	부호가 없음
A>B	JG/JNLE	JA/JNLE
A<B	JL/JNGE	JB/JNAE
A=B	JE/JZ	JE/JZ
A≧B	JGE/JNL	JAE/JNB
A≦B	JLE/JNG	JBE/JNA
A≠B	JNE/JNZ	JNE/JNZ

〔화면 2〕 예제 2의 실행결과 확인

```
C>DEBUG LIST-2.EXE        화일명을 지정하여 DEBUG를 기동
-R        레지스터의 표시                프로그램의 크기
AX=0000  BX=0000  CX=0036  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1056  ES=1056  SS=1066  CS=1066  IP=0000   NV UP EI PL NZ NA PO NC
1066:0000 B86910        MOV     AX,1069        데이터 세그먼트
-
-                          ','를 잘못 입력하여 에러가 발생했다.
-D 1069:0.10
          ^ Error
-                          데이터 세그먼트의 확인
-D 1069:0,10
1069:0000  00 00 00 00 00 00 CC FF-36 CE 5C 8B D8 8A 47 08  ........6.\...G.
1069:0010  2A          초기에는 0으로 되어 있다.                   *
-
-                  프로그램을 실행한다.
-G=0,36         끝번지(CX 레지스터의 값)
     시작번지
Program terminated normally
-
-                         계산 결과
-D 1069:0,10
1069:0000  9F 24 00 00 00 00 CC FF-36 CE 5C 8B D8 8A 47 08  .$......6.\...G.
1069:0010  2A                                                 *
-
-Q

C>
```

말과도 같으므로 'GE'를 'NL'(Not Less than)으로도 표기할 수가 있다. 이들 중 'G'와 'L'은 부호가 있는 수치의 경우에, 'A'와 'B'는 부호가 없는 수치의 경우에 쓰인다.

'Z'은 'Zero'의 의미로 'Equal'과도 일맥 상통한다. 그 이유는 다음과 같다. CMP AX, BX의 경우는 AX와 BX의 값을 비교하라는 의미인데 CPU는 AX와 BX의 크기를 비교하기 위해 AX에서 BX를 뺀 결과가 양수이냐, 음수이냐, 혹은 0이냐를 가지고 대소 여부를 가리게 된다. 이때 0일 경우에는 플래그 레지스터의 Zero 플래그가 1로 세트된다.

제로 플래그가 세트되었다는 의미는 연산의 결과가 0이 되었다는 의미이고, 이것은 결국 AX와 BX의 값이 같다는 의미가 되므로 'Zero'와 'Equal to'를 같은 것으로 취급하게 된다.

플래그 레지스터에 의한 분기

플래그 레지스터는 각각의 비트가 CPU의 상태를 기억한다는 것은 이미 지난 호에서도 밝힌 바 있다. 8088은 플래그 레지스터의 특정 플래그상태에 의해서도 분기가 가능하다. 그 플래그들은 SF, ZF, PF, CF, OF가 있고 이들을 이용한 분기 명령은 **표 2**에 나타낸 바와 같다.

이들 중 CX레지스터는 범용 레지스터인데 8088에는 CX의 값이 0일 때 분기하는 명령이 있다. 이것은 어떤 특정한 부분을 반복 수행할 경우에 반복 횟수를 세는 카운터로 CX를 이용하는 데서 유래한 것이다.

CX에 반복할 횟수를 기억시켜 두고 매번 반복한 후에 1씩 감소하는데 CX의 값이 0이면 다른 곳으로 분기를 하고 그렇지 않으면 반복 처리하는 경우에 쓰이게 된다. 그러면 예제 프로그램을 통해 분기 명령을 실습하기로 한다.

〈예제 4〉

1~100의 합계를 구하라.

〈처리 방법〉

BX레지스터의 값을 1부터 100까지 1씩 증가시켜 가면서 이 값을 계속 AX레지스터에 더해진다. 이때 맨 처음에는 AX의 값을 0으로 클리어시켜 두어야 한다.

〔표 2〕 플래그를 이용한 분기명령

플래그	명　칭	1일때 분기	0일때 분기	플래그가 1일때의 의미
SF	Sign	JS	JNS	최상위 비트가 1이다(음수).
ZF	Zero	JZ/JE	JNZ/JNE	연산 결과가 0이다.
PF	Parity	JP/JPE	JNP/JPO	비트가 1인 갯수가 짝수.
CF	Carry	JB/JNAE	JNB/JAE	캐리 플래그가 1이다(연산결과 자리올림이나 빌림이 발생했다).
OF	Overflow	JO	JNO	연산 결과 오버플로우가 발생했다.
CX	CX 레지스터	—	JCXZ	CX의 값이 0이다.

베이직의 경우에는 초기화하는 과정이 없더라도 변수가 처음 쓰이면 자동적으로 0이 된다. 즉, 다음의 프로그램은 'S=0'이 없더라도 변수 S는 0부터 시작된다.

```
10 N=1
20 S=S+N
30 N=N+1
40 .....
```

그러나 베이직을 제외한 다른 언어에서는 초기화하지 않으면 컴퓨터의 상태에 따라 변수 S의 값에 엉뚱한 값이 기억되게 된다.

어셈블리에서도 처음에 0을 기억시켜두지 않으면 그 전의 AX레지스터의 값이 남아있을 것이므로 여기에 1~100을 더하면 전혀 다른 값이 나오게 된다(리스트 4 참조).

리스트 4의 프로그램을 디버거상에서 실행시킨 후 메모리를 살펴보면 13BAH (5050)이 저장되었음을 발견하게 된다.

1씩 증가와 감소

[형식] INC <오퍼랜드> 또는 DEC <오퍼랜드>

INC는 Increment, DEC는 Decrement의 약어로 각각 증가와 감소를 나타낸다. 이 명령은 레지스터(8/16)나 메모리(8/16)의 내용을 1씩 증가시키거나 1씩 감소시키는 경우에 사용된다. AX레지스터의 내용을 1증가시키는 경우라면 다음과 같다.

```
INC AX
```

이것은 'ADD AX, 1'과 같다.

8088에서는 조건분기 명령이 다양하므로 같은 문제라 하더라도 프로그램을 작성하는 방법이 수없이 많게 된다. 그러므로 어떤 형식이 있어 꼭 이렇게 해야한다 하는 것은 없고, 상황에 따라 적절한 명령을 선택하여 사용하게 된다. 예제 4를 DEC명령과 JCXZ (CX가 0이면 분기) 명령을 이용해 프로그래밍하면 리스트 5가 된다.

〔리스트 3〕 시간의 계산

```
0000                        MAIN    SEGMENT
                                    ASSUME CS:MAIN, DS:DATA
0000  B8 ---- R                     MOV    AX, DATA
0003  8E D8                         MOV    DS, AX
                                    ;  --- Add  Hour
0005  B0 02                         MOV    AL, 2
0007  B3 03                         MOV    BL, 3
0009  02 C3                         ADD    AL, BL
000B  A2 0000 R                     MOV    T1, AL
                                    ;  --- Add Minute ---
000E  B0 2F                         MOV    AL, 47
0010  B3 39                         MOV    BL, 57
0012  02 C3                         ADD    AL, BL
0014  A2 0001 R                     MOV    T2, AL
                                    ;  --- Add Second ---
0017  B0 2D                         MOV    AL, 45
0019  B3 34                         MOV    BL, 52
001B  02 C3                         ADD    AL, BL
                                    ;
001D  B4 00                         MOV    AH, 0
001F  B3 3C                         MOV    BL, 60
0021  F6 F3                         DIV    BL
0023  88 26 0002 R                  MOV    T3, AH
                                    ;
0027  02 06 0001 R                  ADD    AL, T2
002B  B4 00                         MOV    AH, 0
002D  F6 F3                         DIV    BL
002F  88 26 0001 R                  MOV    T2, AH
                                    ;
0033  00 06 0000 R                  ADD    T1, AL
                                    ;
0037  B4 4C                         MOV    AH, 4CH
0039  CD 21                         INT    21H
003B                        MAIN    ENDS
                                    ;
0000                        DATA    SEGMENT
0000  00                      T1    DB  0
0001  00                      T2    DB  0
0002  00                      T3    DB  0
0003                        DATA    ENDS
                                    END
```

계산 결과를 화면에 출력

지금까지는 계산 결과를 아스키 코드로 하는 문자로 화면에 출력하거나 디버거를 이용해서 확인하는 방법을 이용했다. 이제는 계산된 결과를 직접 화면에 출력하는 과정에 대해서 알아보기로 하자.

현재 메모리에 3CH라는 수치가 있

〔화면 3〕 DEBUG를 이용한 실행결과

```
C>DEBUG LIST-3.EXE ←── DEBUG의 기동
-R
AX=0000  BX=0000  CX=0043  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=1056  ES=1056  SS=1066  CS=1066  IP=0000   NV UP EI PL NZ NA PO NC
1066:0000 B86A10        MOV     AX,106A
-
-G=0,43 ←── 프로그램 실행

Program terminated normally
-
-D 106A:0,10 ←── 메모리 확인
106A:0000  06 2D 25 CC 01 00 50 8B-C3 05 09 00 50 E8 E0 53 .-%...P.....P..S
106A:0010  83
-Q          시간  분  초

C>
```

〔리스트 4〕 100까지의 합계 구하기

```
0000                      MAIN    SEGMENT
                                  ASSUME CS:MAIN, DS:DATA
0000  B8 ---- R                   MOV    AX, DATA
0003  8E D8                       MOV    DS, AX
                                  ;
0005  B8 0000                     MOV    AX, 0
0008  BB 0001                     MOV    BX, 1
000B  03 C3               NEXT:   ADD    AX, BX
000D  83 C3 01                    ADD    BX, 1
0010  83 FB 64                    CMP    BX, 100
0013  76 F6                       JBE    NEXT
                                  ;
0015  A3 0000 R                   MOV    ANS, AX
0018  B4 4C                       MOV    AH, 4CH
001A  CD 21                       INT    21H
001C                      MAIN    ENDS
                                  ;
0000                      DATA    SEGMENT
0000  0000                  ANS   DW     0
0002                      DATA    ENDS
                                  END
```

다고 하자. 이 수치는 1바이트의 수치이다. 이것을 화면에 출력하려면 상위 4비트를 '3'이라는 문자로 바꾸어서 출력하고, 또, 하위 4비트를 'C'라는 문자로 바꾸어서 출력하는 과정을 거쳐야 한다. 이 과정은 얼마간은 복잡하지만 MS-DOS에서는 수치를 직접 출력하는 기능이 없으므로 우리가 스스로 만들어야 한다. 이 과정은 다음과 같다.

(1) 이 수치를 AL레지스터에 넣는다.

3	C	
0011	1100	AL

(2) 이 값을 16(10H)으로 나눈 몫과 나머지를 구한다.

이때 나눗셈은 부호를 고려하지 않는 연산(DIV)으로 해야지 IDIV로 연산을 하면 계산 결과가 달라지게 되므로 주의하여야 한다. 또, 나눗셈 명령은 16비트÷8비트=8비트의 몫과 나머지가 구해지므로 AH레지스터의 값을 0으로 해두어야 한다.

0	3	0	C
0000	0011	0000	1100
AL(몫)		AH(나머지)	

이 결과로 상위 4비트가 AL에 하위 4비트가 AH에 저장된다.

(3) 이 값이 0~9 사이의 수치이면 이 값에 30H(48)을 더한다. 그러면 그 값에 해당하는 아스키 문자('0'~'9')가 된다. 왜냐하면 '0'은 30H, '1'은 31H, '2'는 32H..... '9'는 39H 순으로 아스키 코드가 부여되기 때문이다. 이 값이 0AH~0FH사이의 수치이면 이 값에 37H(55)를 더한다. 'A'는 65, 'B'는 66, 'C'는 67..... 'F'는 70으로 아스키 코드가 부여되는데 0AH는 10진수로 10, 0BH는 11, 0CH는 12...이므로 65−10=55(37H)가 되는 셈이다.

문자	0	1	2	3	4	5	…	A	B	C	D	…	
코드	48	49	50	51	52	53	…	65	66	67	68	69	70

(4) 위의 과정으로 만들어진 수치가 바로 원하는 아스키 코드이므로 이것을 DOS 기능호출의 2번 1문자 출력루틴을 이용해서 출력한다. 출력하려는 수치가 2바이트 이상일 경우에는 상위 바이트부터 1바이트씩 나누어서 위의 과정을 반복하면 된다. 그러면 수치를 직접 화면에 출력하는 것을 예제를 가지고 실습해보기로 하자.

〈예제 5〉

2+4+6+8+·····+250의 합을 구하여 그 답을 16진수로 화면에 출력하라.

〈처리 방법〉

덧셈을 하는 과정은 예제 4와 비슷하다. 연산 결과는 15750(3D86H)이므로 2바이트 수치가 된다. 이 값은 메모리에 일단 저장해 두고 1바이트씩 2문자로 바꾸어서 화면에 출력하게 된다. 메모리에 저장될 때는 상위와 하위가 바뀌어서 저장될 것이다(리스트 6 참조).

〈프로그램 설명〉

–0003H : 데이터 세그먼트의 설정
–0014H : 연산
–0016H : 연산 결과를 메모리에 보존 (레지스터의 값에는 변동없음)
–001FH : 상위 바이트를 16으로 나눔 (AL : 몫, AH : 나머지) 몫을 먼저, 나머지를 나중에 출력한다.
–002AH : 몫에 해당하는 수치를 아스키 문자로 바꿈
–0037H : 나머지에 해당하는 수치를 문자로 바꿈
–0042H : 위의 2문자를 화면에 출력
–0044H : 0016H에서 메모리에 저장한 값 중 하위 1바이트를 꺼낸다.
–004BH : 위의 값을 16으로 나눔
–0063H : 위의 결과를 2개의 문자로 변환
–006EH : 위의 2문차를 화면에 출력
–0074H : 프로그램의 종료

위의 리스트에서 보듯이 실제 연산하는 부분보다 문자열로 변환하여 처리하는 과정이 훨씬 더 복잡하고 길다.

〔리스트 5〕 DEC, JCXT명령을 이용한 예

```
0000                        MAIN    SEGMENT
                                    ASSUME CS:MAIN, DS:DATA
0000  B8 ---- R                     MOV    AX, DATA
0003  8E D8                         MOV    DS, AX
                                    ;
0005  B8 0000                       MOV    AX, 0
0008  B9 0064                       MOV    CX, 100
000B  03 C1                 NEXT:   ADD    AX, CX
000D  49                            DEC    CX
000E  E3 02                         JCXZ   QUIT
0010  EB F9                         JMP    NEXT
                                    ;
0012  A3 0000 R             QUIT:   MOV    ANS, AX
0015  B4 4C                         MOV    AH, 4CH
0017  CD 21                         INT    21H
0019                        MAIN    ENDS
                                    ;
0000                        DATA    SEGMENT
0000  0000                    ANS   DW     0
0002                        DATA    ENDS
                                    END
```

앞으로 명령어를 계속 배워가다 보면 좀더 쉽고 간단한 방법을 터득하겠지만 우선은 이정도로 만족하는 수 밖에 없다.

앞에서도 말했듯이 컴퓨터가 탁상용 계산기와 다른 것은 스스로 판단할 수 있다는 점이다. 물론 이 판단은 프로그래머의 판단에 의지한다. 프로그래머가 상황 설정을 잘못해주면 컴퓨터의 처리결과는 당연히 달라진다.

분기명령은 대단히 중요한 것이며, 연산명령과 분기명령만 알고 있어도 어지간한 것들은 처리할 수 있게 된다. 따라서 jump명령은 습관처럼 몸에 배도록 익혀두어야 한다.

다음 호에서는 서브루틴의 작성방법과 반복을 좀더 쉽게 할 수 있는 방법에 대해서 알아보기로 한다. PC

〔리스트 6〕 연산결과를 수치형태로 출력하는 예

```
0000               MAIN   SEGMENT
                          ASSUME CS:MAIN, DS:DATA
0000  B8 ---- R           MOV    AX, DATA
0003  8E D8               MOV    DS, AX
                          ;
0005  B8 0000             MOV    AX, 0
0008  BB 0002             MOV    BX, 2
000B  03 C3        LP:    ADD    AX, BX
000D  83 C3 02            ADD    BX, 2
0010  81 FB 00FA          CMP    BX, 250
0014  76 F5               JBE    LP
0016  A3 0000 R           MOV    ANS, AX
                          ;  ---  CHR 1  ---
0019  8A C4               MOV    AL, AH
001B  B4 00               MOV    AH, 0
001D  B1 10               MOV    CL,  16
001F  F6 F1               DIV    CL
0021  3C 09               CMP    AL, 9
0023  77 05               JA     UPP
0025  04 30               ADD    AL, 30H
0027  EB 03 90            JMP LOW1
002A  04 37        UPP:   ADD    AL, 37H
```

```
002C                LOW1:  ;   ---  CHR 2 ---
002C  80 FC 09             CMP    AH, 9
002F  77 06                JA     UPP1
0031  80 C4 30             ADD    AH, 30H
0034  EB 04 90             JMP    UPP2
0037  80 C4 37      UPP1:  ADD    AH, 37H
                           ;  ---  PRINT OUT ----
003A  8B D0         UPP2:  MOV    DX, AX
003C  B4 02                MOV    AH, 2
003E  CD 21                INT    21H
0040  8A D6                MOV    DL, DH
0042  CD 21                INT    21H
                           ;   ---  CHR 3 ---
0044  A0 0000 R            MOV    AL, BYTE PTR ANS
0047  B4 00                MOV    AH, 0
0049  B1 10                MOV    CL, 16
004B  F6 F1                DIV    CL
                           ;   ---  CHR 4 ---
004D  3C 09                CMP    AL, 9
004F  77 05                JA     UPP3
0051  04 30                ADD    AL, 30H
0053  EB 03 90             JMP LOW2
0056  04 37         UPP3:  ADD    AL, 37H
0058                LOW2:  ;   ---  CHR 2 LOWER BYTE ---
0058  80 FC 09             CMP    AH, 9
005B  77 06                JA     UPP4
005D  80 C4 30             ADD    AH, 30H
0060  EB 04 90             JMP    UPP5
0063  80 C4 37      UPP4:  ADD    AH, 37H
                           ;  --- PRINT OUT ---
0066  8B D0         UPP5:  MOV    DX, AX
0068  B4 02                MOV    AH, 2
006A  CD 21                INT    21H
006C  8A D6                MOV    DL, DH
006E  CD 21                INT    21H
                           ;
0070  B4 4C                MOV    AH, 4CH
0072  CD 21                INT    21H
0074                MAIN   ENDS
                           ;
0000                DATA   SEGMENT
0000  0000           ANS   DW     0
0002                DATA   ENDS
                           END
```

교육용 프로그램

이기성 /장왕교재 연구원 상무

앞에서 우리는 그래픽 명령어와 사운드 명령어를 공부하였다. 이 명령어들을 이용하여 프로그램을 교육용(Computer Aided〈Based, Assisted〉 Education 〈Instruction〉)으로 작성하여 보자. 여기에 소개하는 프로그램은 중앙여고의 성재수씨가 개발한 '물리 학습용 프로그램'에서 필자가 일부를 발췌한 것이다.

시작에 앞서

교육용 프로그램은 그 성격상 그래픽이 화면에 많이 나타나야 하고, 아울러 이 그래픽에 대한 설명이 한글로 나타나야 한다. 또 공부라는 딱딱한 분위기를 벗어나게 하려고, 사운드 명령도 많이 넣어 신나는 음악을 들으면서 프로그램을 진행시켜 나가야 한다. 그러나 우리 나라의 실정에 한글과 그래픽을 동시에 처리할 수 있는 XT 호환 기종의 컴퓨터는 많이 보급되지 않은 상태이다. 컬러모니터를 사용하고, 여기에 한글을 같이 사용하는 독자나, 모노크롬 보드를 내장한 컴퓨터로서 그래픽 화면에서 한글이 바로 나타나는 컴퓨터를 가진 독자는 MS-DOS 상태에서 바로 한글 프로그램을 컴퓨터의 메모리로 올려놓고, BASIC, BASICA, GWBASIC을 사용하면 된다.

세운상가 컴퓨터를 비롯한 대부분의 메이커 제품도 그래픽 화면에서 한글을 사용하기에는 문제가 있다. 한글 사용은 커녕, IBM-XT 호환기종에서 그래픽 명령을 주면 아예 못받아들이는 경우가 대부분이다.(illegal function call 이라는 에러 메시지를 내면서). 허큘리스 카드를 내장한 컴퓨터에서는 GW-BASIC 이나 BASIC 대신 HBASIC을 사용하여야 그래픽 명령을 사용할 수 있는데, 이 HBASIC에서는 한글을 같이 사용하지 못하는 단점이 있다. 삼보컴퓨터에서는 이점을 감안하여 HBASIC 에서도 한글을 사용할 수 있도록 수정하여 GWBASIC이란 이름으로 사용자에게 공급하고 있다.

일반 BASIC 상태에서 그래픽 명령을 사용하려면 허클리스 상태를 컬러모니터 상태로 바꾸어주는 프로그램(Simcga, Magickey)을 DOS 상태에서 미리 실행시켜 주면 가능하다. 그러나 이때도 한글을 사용하지 못하는 점은 마찬가지이다. 그래픽과 한글을 동시에 사용하는 가장 손쉬운 방법은 삼보컴퓨터의 DOS 디스켓중 BASIC 서브디렉토리에 있는 NKP한글과 GW-BASIC을 사용하는 것이라고 필자는 생각한다. 다음에 소개하는 프로그램도 이렇게 해서 작성한 것이다.(삼보컴퓨터의 제품은 NKP를 바로 올려도 괜찮으나 세운상가의 제품은 HGC FULL을 먼저 컴퓨터의 메모리로 올리고 NKP를 실행시키는 것이 좋다)

프로그램의 구성

예제 프로그램은 lksijak.bas와 posache. bas의 2개로 구성된다. lksijak.bas는 화면에 '물리 학습 프로그램'이라고 쓰고는 경쾌한 음악을 들려준다. 음악연주가 끝나면 RUN"posache" 명령으로 자동으로 '포사체와 상승운동 학습' 프로그램으로 넘어간다. posache.bas는 화면에 그래픽을 멋있게 그리고 '공부할 내용'을 화면에 보여준다.

●LKSIJAK.BAS 프로그램

화면에 글자를 쓰는 루틴은 40번행부터 70번행까지이고 음악 소리 루틴은 80행부터 1110행까지이다. 1120행에서 포사체 프로그램을 부른다(리스트 1 참조).

●POSACHE.BAS 프로그램

리스트 2의 20번행에서 320번행까지는 데모용 화면을 그려주는 부분으로 처음에는 '포사체와 상승운동 학습'이라는 메시지가 박스와 함께 나타난다. (140번행의 뒷부분은 컴퓨터가 1부터 1000까지 셀동안 다음행의 실행을 지연시키는 효과를 얻는다.)

```
140 LINE(170,5)-(440,40),7,BF:
    FOR T=1 TO 1000:NEXT T
150 LINE(170,5)-(440,40),0,BF
160 LINE(170,5)-(440,40),7,BF
170 LOCATE 2,27:PRINT"포사체와 상승운동 학습"
```

다음에는 CIRCLE 명령을 이용한 둥근 고리의 입체모양이 나타난다. 프로그램의 앞부분에서 원의 갯수, 작은 원의 반지름, 큰 원의 반지름 및 중심 좌표값을 지정하고 나서 230번행부터는 원의 기울기 정도(E)와 그려지는 위치를 변경시키면서 입체모양을 만드는 원리이다.

〔리스트 1〕 LKSIJAK.BAS

```
10 REM save"lksijak"
20 REM  programmed by Sung Jaisoo
30 REM  edited by Leeks
40 KEY OFF :CLS
50 PRINT "     물리 학습 프로그램    "
60 PRINT : PRINT : PRINT"포사체 운동의 연습입니다. "
70 PRINT : PRINT "음악을 들으면서 잠깐만 기다리세요."
80 PLAY"MBL16T155"
90 FOR I=1 TO 6
100  READ M$
110  PLAY M$
120 NEXT I
130 DATA "o2mnb4p8msbbmnb4p8msbbb8g#8
140 DATA "b8g#8b8o3e8o2mnb4p8msbbmnb4
150 DATA "p8msbbmnb4p8msbbmnb4p8msbb
160 DATA "b8bbb8b8b8bbb8b8b8bb
170 DATA "b8b8b8bbb8b8mlb2
180 DATA "b2b8p8p4p4
190 DATA "p8msolbbb8bbb8bbo2e8f#8g#8olbb
200 DATA "b8bbo2e8g#g#f#8d#8olb8bbb8bb
210 DATA "b8bbo2e8f#8g#8eg#mlb4bmsag#f#
220 DATA "e8g#8e8o3bbb8bbb8bbo4e8f#8
230 DATA "g#8o3bbb8bbo4e8g#g#f#8d#8o3b8bb
240 DATA "b8bbb8bbo4e8f#8g#8mleg#b4
250 DATA "bag#f#mse8g#8e8o3g#g#g#8g#g#g#8g#g#
260 DATA "g#8o4c#8o3g#8o4c#8o3g#8o4c#8o3g#8f#8e8d#8
270 DATA "c#8g#g#g#8g#g#g#8g#g#g#8o4c#8o3g#8o4c#8
280 DATA "o3g#8o4c#8o3b8a#8b8a#8b8g#g#g#8g#g#
290 DATA "g#8g#g#g#8o4c#8o3g#8o4c#8o3g#8o4c#8o3g#8f#8
300 DATA "e8d#8c#8g#g#g#8g#g#g#8g#g#g#8o4c#8
310 DATA "o3g#8o4c#8o3g#8o4c#8o3b8a#8b8o2bbb8f#f#
320 DATA "f#8f#f#f#8g#8a8f#4mna8msg#8mne4
330 DATA "msg#8f#8f#8f#8o3f#f#f#8f#f#f#8g#8
340 DATA "a8mnf#4msa8g#8mne4msg#8f#8o2bb
350 DATA "b8olbbb8bbb8bbo2mne8f#8g#8olbb
360 DATA "b8bbo2e8g#g#f#8d#8olb8bbb8bb
370 DATA "b8bbo2e8f#8g#8eg#mlb4mnbag#f#
380 DATA "e8g#8e8o3bbb8bbb8bbo4e8f#8
390 DATA "g#8o3bbb8bbo4e8g#g#f#8d#8o3b8bb
400 DATA "b8bbb8bbo4e8f#8g#8mleg#mlb4
410 DATA "mnbag#f#mne8g#8e8o3mle56f56g56a56b56o4c56d56mne8eee8
    e8g#4.
420 DATA "f#8e8d#8e8c#8mso3bo4c#o3bo4c#o3b
430 DATA "o4c#d#eo3abababo4c#d#o3g#ag#ag#abo4c#o3f#
440 DATA "g#f#g#f#g#f#g#f#g#f#d#o2bo3mlbo4c#d#e8d#8e8
450 DATA "c#8o3msbo4c#o3bo4c#o3bo4c#d#eo3abababo4c#d#o3g#
460 DATA "ag#ag#abo4c#o3f#g#f#g#f#af#emne8p8mlc#4
470 DATA "mnc#o2cmso3c#o2co3d#c#o2baag#ec#c#c#c#c#e
480 DATA "d#olcg#g#g#g#g#g#o2c#eg#o3c#c#c#c#c#o2co3c#o2co3d#
490 DATA "c#o2baag#ec#c#c#c#c#ed#olcg#g#g#g#g#mng#
500 DATA "o2c#eg#o3msc#ed#c#d#o2cg#g#g#o3g#ec#d#o2cg#g#g#
510 DATA "o3g#ec#d#o2bg#g#a#gd#d#g#gg#gg#ag#f#e
520 DATA "olba#bo2eolbo2f#olbo2g#ed#eg#eaf#bo3g#f#ed#
530 DATA "f#ec#o2bo3c#o2bo3c#d#ef#g#o2ababo3c#d#ef#o2g#
540 DATA "ag#aco3c#d#eo2f#g#f#g#f#g#f#g#f#g#f#d#olb
550 DATA "co2c#d#eolba#bo2eolbo2f#olbo2g#ed#eg#eaf#b
560 DATA "o3g#f#ed#f#ec#o2bo3c#o2bo3c#d#ef#g#o2ababo3c#
570 DATA "d#ef#o2g#ag#abo3c#d#eo2f#o3c#o2co3c#d#c#o2af#mne
580 DATA "o3mlef#g#abo4c#d#mne8mseee8e8g#4.
590 DATA "msf8mse8d#8e8c#8o3bo4c#o3bo4c#o3bo4c#d#eo3a
600 DATA "bababo4c#d#o3g#ag#ag#abo4c#o3f#g#f#g#f#
610 DATA "g#f#g#f#g#f#d#o2bo3mlbo4c#d#mne8eee8e8g#4.
620 DATA "msf#8e8d#8e8c#8o3bo4c#o3bo4c#o3b
630 DATA "o4c#d#eo3abababo4c#d#o3g#ag#ag#abo4c#o3f#
640 DATA "g#f#g#f#ag#f#e8o2b8o3e8g#g#g#8mng#g#g#8
650 DATA "g#g#g#8o4c#8o3g#8o4c#8o3g#8o4c#8o3g#8f#8e8
660 DATA "d#8c#8g#g#g#8g#g#g#8g#g#g#8o4c#8o3g#8
670 DATA "o4c#8o3g#8o4c#8o3b8a#8b8a#8b8g#g#g#8
680 DATA "g#g#g#8g#g#g#8o4c#8o3g#8o4c#8o3g#8o4c#8o3g#8
690 DATA "f#8e8d#8c#8g#g#g#8g#g#g#8g#g#g#8
700 DATA "o4c#8o3g#8o4c#8o3g#8o4c#8o3b8a#8b8a#8b8
710 DATA "o2f#f#f#8f#f#f#8g#8a8f#4a8g#8
720 DATA "e4g#8f#8o0b8olb8o2f#f#f#8f#f#f#8
730 DATA "g#8a8f#4a8g#8e4g#8f#8
740 DATA "bbb8olbbb8bbb8bbo2e8f#8g#8
750 DATA "olbbb8bbo2e8g#g#f#8d#8olb8bbb8
760 DATA "bbb8bbo2e8f#8g#8eg#mlb4mnb
770 DATA "ag#f#e8olb8o2e8o3bbb8bbb8bbo4e8
780 DATA "f#8g#8o3bbb8bbo4e8g#g#f#8d#8o3b8
790 DATA "bbb8bbb8bbo4e8f#8g#8o3eg#mlb4
800 DATA "mnbag#f#mlef#g#mnamlg#abo4mnc#mlo3bo4c#d#mnemld#
810 DATA "ef#mng#ao3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3
    bmle
820 DATA "f#g#mnamlg#abmno4c#mlo3bo4c#d#mnemld#ef#mng#ao3bo4ao
    3bo4a
830 DATA "o3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3bp16mlg#o4g#o3mng#p16
    mld#o4d#o3mnd#p16
840 DATA "mleo4eo3mnep16mlao4ao3mnap16mlg#o4g#o3mng#p16mld#o4d
    #o3mnd#p16mleo4eo3mnep16
850 DATA "mlao4ao3mnao4go3go4go3go4go3go4go3go4msg8e8c8e8o4mng#
860 DATA "o3g#o4g#o3g#o4g#o3g#o4g#o3g#o4msg#8e8o3b8o4e8mng#o3g#
    o4g#o3g#o4g#
870 DATA "o3g#o4g#o3g#o4msg#8f8c#8f8mna#o3a#o4a#o3a#o4a#o3a#o4a
    #o3a#o4msa#8
880 DATA "g8e8g8b8p16mna#p16ap16g#p16f#p16ep16
890 DATA "d#p16c#p16o3bp16a#p16ap16g#p16f#p16ep16d#p16f#mle
900 DATA "f#g#mnamlg#abmno4c#o3mlbo4c#d#mnemld#ef#mng#ao3bo4ao3
    bo4a
910 DATA "o3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3bmlef#g#mnamlg#abmno4
    c#o3mlb
920 DATA "o4c#d#mnemld#ef#mng#ao3bo4ao3bo4ao3bo4ao3bo4ao3bo4ao3
    bo4a
```

```
930 DATA "o3bo4ao3bp16mlg#o4g#o3mng#p16mld#o4d#o3mnd#p16mleo4eo
    3mnep16mlao4ao3mnap16
940 DATA "mlg#o4g#o3mng#p16mld#o4d#o3mnd#p16mleo4eo3mnep16mlao4
    ao3mnao4go3go4go3go4g
950 DATA "o3go4go3go4g8e8c8e8g#o3g#o4g#o3g#o4g#o3g#o4g#o3g#o4g#8
960 DATA "e8o3b8o4e8g#o3g#o4g#o3g#o4g#o3g#o4g#o3g#o4msg#8mnf8c#8
970 DATA "f8a#o3a#o4a#o3a#o4a#o3a#o4a#o3a#o4a#8g8e8g8b8
980 DATA "p16a#p16ap16g#p16f#p16ep16d#p16c#p16o3bp16a#p16
990 DATA "ap16g#p16f#p16ep16d#p16fmled#ed#mne8bbb8
1000 DATA "bbb8bbo4e8f#8g#8o3bbb8bbb8
1010 DATA "bbo4g#8a8b8p8e8f#8g#8p8o3g#8
1020 DATA "a8b8p8p2o2bco3c#dd#
1030 DATA "eff#gg#aa#bco4c#d#ed#f#d#ed#f#d#e
1040 DATA "d#f#d#ed#f#d#ed#f#d#ed#f#d#ed#f#d#e
1050 DATA "d#f#d#e8eo3eo4eo3eo4eo3eo4e8o3bo2bo3bo2bo3bo2bo3b8
1060 DATA "g#o2g#o3g#o2g#o3g#o2g#o3g8eo2eo3eo2eo3eo2eo3e8eee8
1070 DATA "e8e8o2bbb8b8b8g#g#g#8g#8g#8
1080 DATA "eee8e8e8o1b8o2e8o1b8o2g#8e8b8
1090 DATA "g#8o3e8o2b8o3e8o2b8o3g#8e8b8g#8o4e4
1100 DATA "p8eee8e8e8e8e4p8.
1110 DATA "ee4p8.o2ee2
1120 RUN "posache"
1130 END
```

〔리스트 2〕 POSACHE.BAS

```
10 REM save"posache"
20 KEY OFF :SCREEN 2 :CLS :TPI=8*ATN(1)
30 REM--------------------- circle -------------------
40 NR=47 '원의갯수
50 MR=55'작은원반지름
60 LR=220'큰원의 반지름
70 TCX=320' x중심점
80 TCY=180'  y중심점
90 TLT=TPI/20' 표제크기
100 FOR P=0 TO 3 :GOSUB 130  :NEXT P :P=4
110 SCREEN ,,P,P : FOR N=1 TO 40 : NEXT :P=P+1
120 IF P<8 THEN 110 ELSE 140
130 O=P*TPI/(NR*4)  : CLS
140 LINE (170,5 )-(440,40),7,BF : FOR T=1 TO 1000 :NEXT
150 LINE (170,5 )-(440,40),0,BF
160 LINE (170,5 )-(440,40),7,B
170 LOCATE 2,27: PRINT "포사체 와   상승운동 학습 " : FOR T=1 TO
    3000:NEXT
180 LINE (170,5 )-(440,40),7,BF : FOR T=1 TO 1000 :NEXT
190 LINE (170,5 )-(440,40),0,BF
200 LINE (170,5 )-(440,40),7,B
210 LOCATE 2,27: PRINT "포사체 와   상승운동 학습 "
220 LOCATE 23,19:PRINT "잠시 기다리시면 공부할 내용이 나타납니다!!!"
230 FOR N=O TO O+TPI STEP TPI/NR
240 X=LR*COS(N)+TCX
250 Y=LR*TLT*SIN(N)+TCY
260 E=ABS((2/3)/COS(N))
270 IF E>1 THEN R=MR ELSE R=MR/E
280 CIRCLE(X,Y),R,1,,,E  :NEXT N
290 FOR N=0 TO TPI STEP TPI/16
300 R=LR+MR*COS(N)*3/2  : X=TCX
310 Y=TCY+MR*SIN(N)*.87 :E=TLT
320 CIRCLE(X,Y),R,1,,,E  :NEXT N
330 REM ----------------- melody---------------------------
340 SCREEN 2,0:          KEY OFF
350 DD$ = "O2DO3DO2DO3D":BB1$ = "O3DE-DC+": BB2$ = "O3GG-FEE-
    A-GF+GF+FED+EF+"
360 BB1$ = "O3DE-DC+": BB2$ = "O3GG-FEE-A-GF+GF+FED+EF+"
370 RESTORE 880              'SET UP FOR MUSIC
380 READ PL$: PLAY PL$
390 FOR I = 1 TO 50
400     FOR J = 1 TO 10: NEXT
410 NEXT
420 READ PL$: PLAY PL$
430 FOR I = 1 TO 51
440     FOR J = 1 TO 10: NEXT
450 NEXT
460 X = 280
470 READ PL$: PLAY PL$
480 FOR A = 0 TO 3.4 STEP .2
490   Y = 5*(COS(2*A) + SIN(A))+60: X = X-4
500 NEXT
510 READ PL$:PLAY PL$:
520 FOR A = 3.6 TO 6.2 STEP .2
530   Y =15*(COS(2*A) + SIN(A))+60: X = X-4
540 NEXT
550 READ PL$: PLAY PL$
560 FOR A = 1 TO 5.4 STEP .2
570     Y = 30*COS(A) + 60
580 NEXT
590 READ PL$: PLAY PL$
600 FOR A = 6.55 TO 9.399999 STEP .2
610     X = 40 * SIN(A) + 160: Y = 30 * COS(A)+ 60
620 NEXT
630 READ PL$: PLAY PL$
640 FOR A = 3 TO 8 STEP .4
650     X = 30*COS(3*A) +150: Y = 10*A
660 NEXT
670 READ PL$: PLAY PL$
680 FOR A = 3 TO 6.4 STEP .2
690     X = 60 * SIN(A) + 200: Y = 20 * COS(A)+ 40
700 NEXT
```

```
710 READ PL$: PLAY PL$
720 FOR A =6.6 TO 9.399999 STEP .2
730    X = 60 * SIN(A) + 200: Y = 20 * COS(A)+ 40
740 NEXT
750 READ PL$: PLAY PL$
760 FOR I = 1 TO 40
770    X = X - 4: Y = Y +1
780 NEXT
790 READ PL$: PLAY PL$: X = 145: Y = 78
800 FOR Y = 80  TO 40  STEP -2
810 NEXT
820 X1 = X: Y1 = Y
830 READ PL$: PLAY PL$: X = 41: Y = 60
840 FOR X1= X1 TO 9 STEP -2
850    Y1= Y1-.5
860 READ PL$: PLAY PL$: X = 145: Y = 78 : GOTO 1000
870 NEXT
880 DATA "MB MN L16 T150"
890 DATA "XDD$; XDD$;XDD$; XDD$;","XDD$; XDD$; XDD$; XDD$;"
900 DATA "XBB1$; XBB1$; XBB1$; XBB1$;","XBB1$; XBB1$; XBB1$;
    XBB1$;"
910 DATA "O3DD-CO2BB-O3E-DC+DC+CO2BA+BO3CC+"
920 DATA "O3DD-CO2BB-O3E-DC+DC+CO2BA+BO3CC+"
930 DATA "O3DD-CO2BO3CO2BB-AB-BO3CD-DE-DC+D1"
940 DATA "XBB2$;"," XBB2$;"
950 DATA "O3GG-FEFEE-DE-EFG-GA-GF+G1"
960 DATA "O2AB-AG+B-G+B-G+A8B8O3C8D8"
970 DATA "EFED+FD+FD+E8F+8G8G+8A8P8"
980 DATA "O2EFF+GG+AB-B O3CC+DD+","EFF+GG+AB-B O4CC+DD+EF+GG+"
990 DATA "A8P8P4 O3A8P8P4 O1A8"
1000 REM ------------------- title ------------------------
1010 SCREEN 0,0,0 :CLS
1020 COLOR 0,7:LOCATE 4,24:PRINT"   #####   공부할 내용   #
    ####   "
1030 LOCATE 8,10 :PRINT"  1 * 지면에서 어느각도로 던진 물체는
    어떠한 운동을 하는가 ?   "
1040 LOCATE 11,10:PRINT"  2 * 몇도로 던진 물체가 가장 높이 올라가는가 ?
    "
1050 LOCATE 14,10:PRINT"  3 * 몇도로 던진 물체가 가장 멀리가서
    떨어지는가 ?           "
1060 LOCATE 17,10:PRINT"  4 * 어떤 각도끼리 같은 위치에 떨어 지겠는가 ?
    "
1070 LOCATE 21,3 :PRINT"
                "
1080 LOCATE 22,3 :PRINT"  공부할 내용을 이해한 학생은  RETURN  KEY를
    누르시면 다음으로 진행 합니다 "
1090 LOCATE 23,3 :PRINT"
                ": COLOR 7,0
1100 FOR T=1 TO 5000 : NEXT T
1110 LOCATE 24,50 : PRINT" 이해하셨읍니까";
1120 A$=INKEY$:INPUT A$ : IF A$=" " THEN 1120 ELSE 1130
1130 SCREEN 2    :CLS
1140 LINE( 20, 20)-(630,390),7,B
1150 LINE( 60, 50)-(590,360),7,B
1160 PAINT (21,21),7
1170 PAINT (25,25),0
1180 LINE( 20, 20)-(630,390),7,B
1190 LINE( 60, 50)-(590,360),7,B
1200 PAINT (21,21),7
1210 X0=0 :Y0=0 :V0=60  :G=9.8 :M=1 : LINE(61,51)-(589,359),
    0,BF
1220 FX=0 :FY=-M*G :DT=.2  : K=.7 :L=2.5
1230 LINE(130,90 )-(130,275) ,7
1240 LINE(130,275)-(510,275) ,7
1250 LOCATE 5,15:PRINT"높이" : LOCATE 17,60 :PRINT"거리"
1260 LOCATE 3,25 :PRINT "던지는 각도에 따라 떨어지는 위치 비교"
1270 LOCATE 5,35 :PRINT "RETURN KEY를 누르면 진행 합니다  ";
1280 LOCATE 6,45 :PRINT " 진행 할까요 ";
1290 B$=INKEY$ :INPUT B$
1300 LOCATE 6,45 : PRINT "                         ":  IF B$=" " THEN
    1290 ELSE 1310 ,1330 ,1350 ,1370 ,1390
1310 Q=20 :GOSUB 1410 :LOCATE 19,18:PRINT"20도로 던진 물체 였읍니다"
1320 LOCATE 17,26:PRINT"20'"
1330 Q=20 :GOSUB 1410 :LOCATE 20,18:PRINT"30도로 던진 물체 였읍니다"
1340 LOCATE 15,30:PRINT"30'"
1350 Q=30 :GOSUB 1410 :LOCATE 21,18:PRINT"45도로 던진 물체 였읍니다"
1360 LOCATE 11,32:PRINT"45'"
1370 Q=45 :GOSUB 1410 :LOCATE 22,18:PRINT"60도로 던진 물체 였읍니다 "
1380 LOCATE  8,30:PRINT"60'"
1390 Q=60 :GOSUB 1410 :LOCATE 19,45 :PRINT"계속 진행할까요(RETURN)";
1400 C$=INKEY$ :INPUT C$: IF C$=" " THEN 1400 ELSE 1530
1410 X=X0 :Y=Y0
1420 VX=V0*COS(Q*3.14/180)   :VY=V0*SIN(Q*3.14/180)
1430 AX=FX/M  : AY=FY/M
1440 VX=VX+AX*(DT/2)  :VY=VY+AY*(DT/2)
1450 X=X+VX*DT  :  Y=Y+VY*(DT/2)
1460 AX=FM/M  :AY=FY/M
1470 VX=VX+AX*DT  :VY=VY+AY*DT
1480 X1=X*K  : Y1=-Y*L+185
1490 IF X1>=279 OR Y1<=5 OR Y1>=185 THEN 1520
1500 PSET(X1+131,Y1+91),7
1510 GOTO 1450
1520 RETURN
1530 LOCATE 19,45:PRINT" 다음 학습은 무엇 입니까 "
1540 LOCATE 20,47:PRINT" 1 번    다시보여줌 "
1550 LOCATE 21,47:PRINT" 2 번    원하는 각도로던짐 "
1560 LOCATE 22,47:PRINT" 3 번    학습 정리  " ;
1570 LOCATE 19,70:INPUT N
1580 ON N GOTO  1210 ,1590 ,2000
1590 LINE (65,55)-(585,355),7,BF
1600 LINE (65,55)-(585,355),0,BF
1610 LOCATE 3,25 :PRINT "던지는 각도에 따라 떨어지는 위치 비교"
1620 LOCATE 5,25 :PRINT "학생이 원하는 임의의 각도로 던지겠읍니다   ",
1630 LINE(130,90 )-(130,275) ,7
```

```
1640 LINE(130,275)-(510,275) ,7
1650 LOCATE 5,15:PRINT"높이" : LOCATE 17,60 :PRINT"거리"
1660 LOCATE 3,25 :PRINT "던지는 각도에 따라 떨어지는 위치 비교"
1670 LOCATE 19,18:INPUT"원하는 각도는 몇도 입니까 " ,Q
1680 LOCATE 19,18 :PRINT"                                던진각도 =
     "; Q   "도" ;
1690 LOCATE 10,45 :PRINT"던진각도 = "; Q   "도" :LOCATE 21,18:
     PRINT"계속진행 (RETURN)   ";
1700 X0=0 :Y0=0 :V0=60  :G=9.8 :M=1
1710 FX=0 :FY=-M*G :DT=.2  : K=.7 :L=2.5
1720 Q=Q: GOSUB 1740
1730 D$=INKEY$ :INPUT D$: IF D$=" " THEN 1730 ELSE 1860
1740 X=X0 :Y=Y0
1750 VX=V0*COS(Q*3.14/180)   :VY=V0*SIN(Q*3.14/180)
1760 AX=FX/M  : AY=FY/M
1770 VX=VX+AX*(DT/2)  :VY=VY+AY*(DT/2)
1780 X=X+VX*DT  :  Y=Y+VY*(DT/2)
1790 AX=FM/M  :AY=FY/M
1800 VX=VX+AX*DT  :VY=VY+AY*DT
1810 X1=X*K  : Y1=-Y*L+185
1820 IF X1>=279 OR Y1<=5 OR Y1>=185 THEN 1850
1830 PSET(X1+131,Y1+91),7
1840 GOTO 1780
1850 RETURN
1860 LOCATE 21,18:PRINT"또 해보시겠읍니까(Y/n)";
1870 INPUT " ",C$
1880 IF C$="Y" OR  C$= "y" THEN 1890  ELSE 1900
1890 LOCATE 19,18:PRINT"                               " :
     LOCATE 21,18:PRINT"                               " : GOTO 1670
1900 IF C$="N" OR  C$="n" THEN 1910
1910 LINE (65,280)-(582,355),7,BF
1920 FOR T=1 TO 500:NEXT
1930 LINE (65,280)-(582,355),0,BF
1940 FOR T=1 TO 500:NEXT
1950 LINE (65,280)-(582,355),7,B
1960 FOR T=1 TO 500:NEXT
1970 LOCATE 19,15 :PRINT"지금까지 우리들이 실제로 해볼수 없는 실험을
     COMPUTER를"
1980 LOCATE 20,15 :PRINT"통해서 관찰 하였읍니다 .  이해할 수 있는지요 "
1990 LOCATE 22,20 :PRINT"RETURN KEY를 누르면 학습정리로 진행 합니다 .";
2000 D$=INKEY$ :            IF D$=" " THEN 2000 ELSE 2010
2010 LINE( 61, 51)-(589,359),0,BF
2020 LOCATE 3,23 : PRINT "  ***포사체와 상승운동의 학습정리 ***  "
2030 LOCATE 6,12: PRINT "*** 학습 정리 ***"
2040 LOCATE 7,15 :PRINT"지금까지 포사체와 상승운동에 대해서 공부를하였읍니다
     ."
2050 LOCATE 8,15 :PRINT"도움이 많이 되었는지요 ? "
2060 LOCATE 9,15 :PRINT"방금 공부한 내용을 정리 해보겠 읍니다."
2070 LOCATE 10,15:PRINT"10도에서 45도 까지는 각도가 커지면 물체는 멀리
     나간다"
2080 LOCATE 11,15:PRINT"45도가 넘을때는 물체가 오히려 가까이 떨어진다 "
2090 LOCATE 12,15:PRINT"포물선 운동에서 45도의 방향으로 던진물체가 가장멀리
     나간다"
2100 LOCATE 14,12:PRINT"*** 학습 안내 ***"
2110 LOCATE 15,15:PRINT"던지는 부분만 다시 공부하려면  1 번을 누르세요."
2120 LOCATE 17,15:PRINT"포사체와 상승운동 단원을  다시 공부하려면  2 번을
     누르세요."
2140 LOCATE 21,15:PRINT"C.A.I. 물리학습을 마치려면  4 번을 누르세요."
2150 LOCATE 22,40:INPUT"원하는 번호를  누르세요. ",B
2160 ON B GOTO  1210,2170,2180,2200
2170 GOTO 10
2180 RUN "lksijak"
2200 END
```

```
240 X=LR*COS(N)+TCX
250 Y=LR*TLT*SIN(N)+TCY
260 E=ABS((2/3)/COS(N))
280 CIRCLE(X,Y),R,1,,,E
```

원이 그려지는 중심점의 변화를 얻기 위해 사인함수와 코사인함수를 통하여 X좌표는 점차로 작아지다가 커지고 Y좌표는 점차로 커지다가 작아지는 효과를 얻어 타원모양의 고리영역을 만들고 있는데 여기서 원의 갯수와 반지름(NR, MR) 등을 변경하면 여러가지 효과를 볼 수 있을 것이다(화면 1 참조).

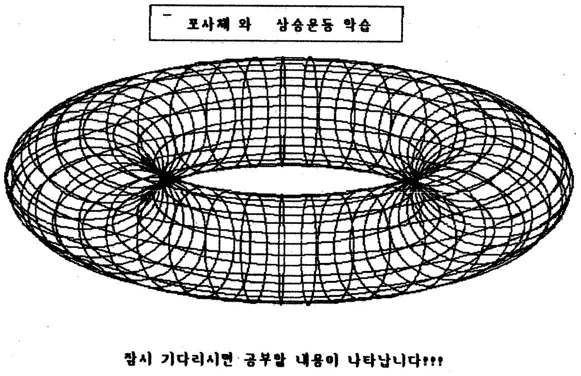
이 화면은 〈Shift〉-〈PrtSc〉키로 프린터에서 인쇄한 것으로 삼보용 NKP와 GWBASIC을 사용하는 경우에는 베이직을 사용하기 전에 미리 'PRTSC S24'라는 명령을 프린터에 보내야만 그림과 문자가 같이 인쇄한다. 만일 이 명령을 생략해 버리면 화면의 그림은 인쇄되지 않고 문자만 인쇄된다. 또 프로그램 중에서 사용하는 'SCREEN 2' 명령은 컴퓨터에 따라 실행되지 않을 수도 있으므로 독자들이 갖고 있는 컴퓨터의 그래픽화면 명령을 여러가지로 테스트해 보고 적합한 화면을 선택하여야 한다. (특히 SCREEN 명령이 컴퓨터의 기종에 따라 다르게 쓰이고 있음을 볼 수 있다.)

다음, 330번 행부터는 중간 멜로디와 시간지연루틴 등으로 이루어져 있고 1000번 행부터는 물리학습에 관한 내용들이다. 1010행을 보면 텍스트모드로 지정하고 공부할 내용을 화면에 출력한 후 키의 입력을 기다리게 된다.

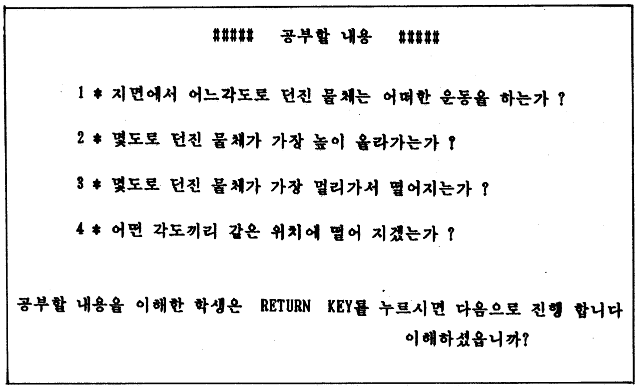
```
1110 LOCATE 24,50: PRINT"이해하셨읍니까";
1120 A$=INKEY$:INPUT A$: IF A$= " " THEN
     1120 ELSE 1130
```

1020행부터 1110행까지가 공부할 내용의 안내문(화면 2 참조)이고 1290행과 1400행은 INKEY$로 한바이트를 키보드로부터 받아들이며 1530행부터 1580행이 '다음 학습 내용'을 1, 2, 3번 중에서 선택하는 루틴이다(화면 3 참조).

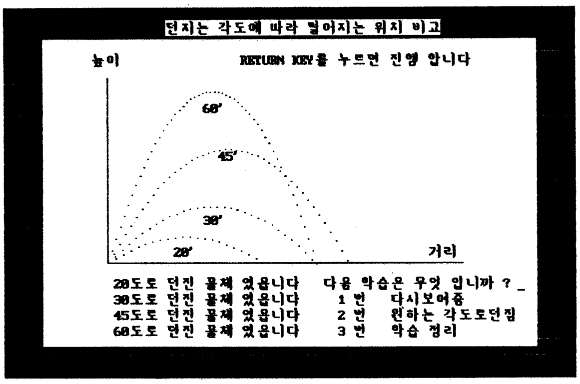
〔화면 1〕 포사체 시작 화면



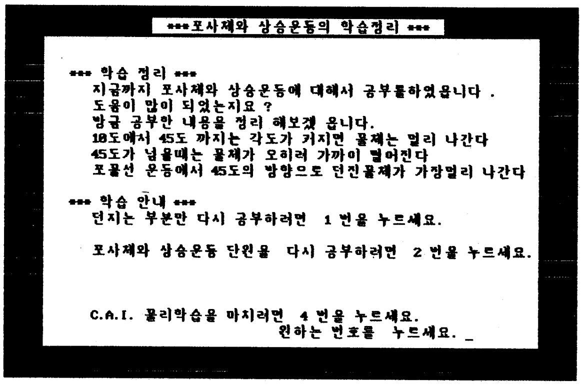
〔화면 2〕 학습할 내용



〔화면 3〕 던지는 각도에 따른 위치 비교



〔화면 4〕 학습정리



1580행에서는 원하는 루틴으로 제어를 옮겨주고 2020행에서는 '학습정리'를 2100행에서는 '학습안내'를 한다(화면 4 참조).

```
1290 B$=INKEY$: INPUT B$
1300 IF B$=" " THEN 1290 ELSE 1310,1330,
     1350, 1370,1390
  :       :
```

```
1400 C$=INKEY$: INPUT C$
1580 ON N GOTO 1210, 1590, 2000
```

포사체의 각도에 따른 운동을 묘사해 주는 부분은 1410행부터 시작되는 루틴인데 점을 찍을 좌표를 계산한 후 PSET 명령으로 출력시키고 있다.

2150번에서 원하는 번호를 입력받는데, 끝내려면 4를 선택한다.

```
2150 LOCATE 22,40:INPUT"원하는 번호를 누르세요.
     ",B
2160 ON B GOTO 1210, 2170,2180,2200
2170 GOTO 10
2180 RUN "lksijak"
2200 END
```

시스템 콜

이용일/한국전자통신연구소

시스템과 프로세스 또는 프로그램의 인터페이스로 지난호에서 설명한 PSP와 함께, 시스템 콜을 들 수 있다. PSP는 프로세스를 호출하는 것으로서 친프로세스와 자프로세스의 인터페이스이며, DOS는 그 중개를 하는 것이라고 이해할 수 있다. 그것에 대하여 시스템 콜은 디스크 장치를 비롯한 여러가지 자원과 프로세스 사이의 가교이며, DOS와 프로세스사이의 인터페이스로서 핵심적인 것이라고 말할 수 있다. MS-DOS에서는 이 시스템 콜의 루틴에서 하드웨어의 차이를 흡수하고, 여러가지 자원을 프로세스에 제공하고 있다.

MS-DOS에서는 기본적으로 시스템 콜을 이용하기 위해 프로세서의 INT 명령을 이용하고 있기 때문에 이번 회에서는 시스템 콜과 함께, 인터럽트 벡터에 대해 최소한으로 필요한 정보를 설명한다.

인터럽트 벡터의 할당

8086프로세서에서는 INT명령어를 사용하여 소프트웨어적으로, 혹은 하드웨어적으로, 00H~FFH까지의 256개의 인터럽트를 사용할 수 있다. 이 중에서 00H~1FH까지의 벡터는, 8086 프로세서의 개발회사인 인텔(Intel)사에 의해 예약되어 있어, 일반 사용자는 사용할 수 없게 되어 있다. 이 범위의 벡터는 정의되지 않은 명령어 및 제로(0)로 나누기(Zero Devide)를 실행하려 한 경우 등, 프로세서에 의해 검출되는 인터럽트용의 벡터로 되어 있다.

MS-DOS에서는, 8086프로세서의 일반 사용자에게 개방되어 있는 벡터 중 20H~3FH를 시스템용으로서 예약하고 있다. 따라서, MS-DOS상에서 프로그램을 개발하는 사용자가 자유롭게 이용할 수 있는 것은 40H~FFH범위의 벡터이고, 00H~3FH사이의 벡터를 프로그램 내부에서 새로 정의하여 사용하는 것은 금지되어 있다. 또 기종에 따라서는 사용자에게 개방되어 있는 벡터의 일부를 하드웨어 회사가 예약하고 있는 경우도 있기 때문에, 그 벡터도 사용자는 사용할 수 없다. 표1에 인터럽트 벡터의 할당을 보여주고 있다.

MS-DOS의 시스템 콜

앞에서 설명한 것처럼, MS-DOS에서 시스템 콜용의 엔트리로서 준비되어 있는 것은 20~3FH까지의 벡터이지만, 실제로는 그들 모두가 사용되고 있는 것은 아니고, 반수 이상의 벡터가 장래의 확장을 위해 예약되어 있는 것에 불과하다. 또 현재 준비되어 있는 시스템 콜의 대부분은 CP/M 및 MS-DOS 초기 버전과의 호환성을 갖기 위해 준비되어 있는 것과, 특수한 처리를 하는 것이 있으며, 실제의 프로그램 작성에 사용하는 시스템 콜은 극히 일부이다. 보다 자세한 내용은 다음과 같다.

(1) INT 20H

이 시스템 콜(INT 20H : 프로세스의 종료)은 프로세스가 끝났을 때에 실행되어, 제어를 시스템으로 돌린다. 즉 프로세스를 종료시키는 것이다. 그러나 이 시스템 콜을 이용하는 것은 그다지 권고하고 싶지 않다. 그 이유는, 펑션 콜(다음 항에서 설명할 것이다)에도 프로세스를 종료하기 위한 기능이 준비되어 있으며, 기능면에서 보아도 펑션 콜 쪽이 리턴코드를 넘길 수 있어 더 풍부하고 높으며, 마이크로소프트사도 프로세스의 종료에는 펑션 콜을 사용할 것을 권장하고 있기 때문이다. 왜냐하면, 이 시스템 콜은 초기 MS-DOS와의 호환성을 갖기 위해서 준비되어 있기 때문이다.

(2) INT 21H

이 시스템 콜(INT 21H : 펑션 콜)은, MS-DOS가 프로세스에 제공하는 여러가지 다양한 기능을 이용하기 위한 것이다. MS-DOS에서 프로그램을 작성하는 경우, 보통은 이 펑션 콜을 사용하는 것으로 충분하다.

또, 본 연재에서는 INT XXH에 의해 기동되는 시스템 콜과 INT 21H에 의해 기동되는 각 기능과의 혼란을 피하기 위해, INT 21H의 각각의 기능을 '펑션 콜'이라고 부르기로 한다. 또, 펑션 콜에서는 호출할 때의 AH레지스터의 값에 따라, 여러가지 기능이 선택되는데, AH레지스터의 값 X에 의해 호출된 기능을 '펑션 X'라고 부른다. 예를 들어, AH에 44H를 격납하고 펑션 콜을 수행하면 펑션 44H가 실행된다고 한다.

(3) INT 22H, INT 23H, INT 24H

이들 3개의 벡터(INT 22H, INT 23H, INT 24H : 에러 등의 처리루틴으로의 포인터)는 정확히 말해서 INT 명령어에 의해 기동되는 루틴을 가리키는 것이 아니라 프로세스의 종료시 및 치명적인 에러가 발생했을 때, 또는 사용자에 의한 처리중단 요구인 'CTRL-Z'가 입력되었을 경우의 에러 핸들러(처리루틴)의 엔트리 주소를 격납하고 있는 것에 불과하다. 따라서, 이들 벡터를 INT명령어로 기동하면, 예기치 못한, 때로는 치명적인 결과를 내게 된다.

이들 벡터는, 그 내용을 변경하는 펑션 25H에 의해 사용자의 처리중단 요구(User Abort)를 포함한 각종 처리중단시에 프로세스의 마무리를 하기 위해 이용된다.

(4) INT 25H, INT 26H

펑션 콜에 의한 디스크의 액세스는, 모두 MS-DOS의 양식(Format)에 따라서 MS-DOS 시스템이 각종의 처리를 하고나서 프로그램으로 넘겨지기 때문에, 일반적인 화일 이외의 데이터, 예를 들면 디스크의 관리영역 및 디렉토리 등은 직접 액세스할 수 없다. 그것에 대하여, 이 2개의 시스템 콜(INT 25H, INT 26H : 절대 디스크 읽기/쓰기)은, MS-DOS의 양식에 따른 디스크이면, 디스크 상의 임의의 데이터에 액세스할 수 있기 때문에, 디렉토리 및 FAT 등, 보통의 펑션 콜에서는 읽어낼 수 없는 데이터를 자유롭게 액세스할 수 있게 된다. MS-DOS의 표준 명령어인 CHKDSK 및 RECOVER 등은 이 시스템 콜을 이용하여 디스크의 관리정보를 프로그램에서 읽어들여, 그 기능을 실현하고 있는 것이다. 이 2개의 시스템 콜의 이용방법에 대해서는 나중에 설명하겠다.

(5) INT 27H

이 시스템 콜(INT 27H : 프로세스의 상주종료)은 INT 20H와 마찬가지로 프로세스를 종료하기 위한 시스템 콜이지만 INT 20H와 달리 프로세스가 확보한 메모리(영역)를 해방시키지 않고 그대로 메모리에 상주시킨 채로 종료한다. 이것에 의해, MS-DOS의 PRINT 명령어와 같이 메모리에 상주하는 프로그램을 작성할 수 있게 된다. 그러나, INT 20H와 펑션 콜의 관계는 여기서는 성립한다. 즉, 이 시스템 콜의 존재이유는 과거의 MS-DOS와의 호환성을 갖기 위한 것일 뿐이므로, 이 기능을 이용하고 싶을 경우에는 펑션 31H인 펑션 콜의 이용을 권장한다.

(6) INT 2AH

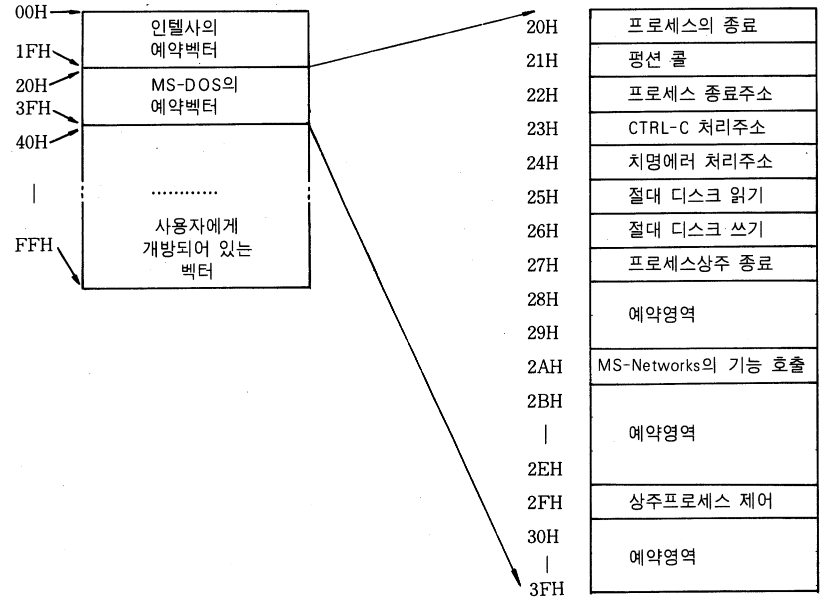
이 시스템 콜(INT 2AH : MS-네트워크 기능의 호출)은, 버전 3.1이상에서만 사용가능한 MS-Network을 짜넣을 때, Network독자의 기능을 호출하기 위해 사용한다. 본 연재에서는 MS-Network는 취급하지 않기 때문에 설명은 생략하지만, 이 시스템 콜을 사용하면, Network 독자적인 프로그램을 기술할 수 있게 된다.

(7) INT 2FH

이 인터럽트(INT 2FH : 상주 프로세스의 제어(버전 3.1이후))는 엄밀히 말하면 MS-DOS에 의해 처리되는 시스템 콜이 아니라 메모리에 상주한 프린트스풀러(Print Spooler) 및 사용자가 작성한 프로세스를 일반적인 명령어 등의 프로세서에서 제어하기 위해 사용할 수 있다. 본 연재에서는 이 기능도 취급하지 않지만 그 기능에 대해서 설명은 해둔다.

이전에는 상주 프로세스와 일반 프로세스 혹은 상주 프로세스 사이의 통신을 수행하는 데는 사용자에게 개방된 인터럽트 벡터를 사용할 수 밖에 없었다. 그러나 이 방법을 사용하면 인터럽트 벡터라는 자원이 상주 프로세스의 수 만큼 소비될 뿐만 아니라 2개의 상주 프로세스가 이용하는 인터럽트 벡터가 충돌할 경우에 한쪽의 벡터를 변경하는 것이 매우 어려워져 버린다. 이 시스템 콜은 그와 같은 결점을 보충하기 위해 추가된 기능이고, 상주 프로세스와의 통신을, 하나의 벡터로 수행하려한 것이다. 이 방법에서는 각 상주 프로세스에 하나의 통신용 번호

〔표 1〕 인터럽트 벡터의 할당



(0~255)를 할당하고, 일반의 프로세스가 상주 프로세스를 호출한 경우에는 그 번호를 기능 코드와 함께 넘기는 것에 의하여 하나의 벡터로 복수의 상주 프로세스를 제어할 수 있도록 고안한 것이다. 번호를 붙이는 것은 프로그램 작성자이고 번호의 충돌이 일어나는 것에 관해서는 종래의 방법과 큰 차이가 없지만 상주 프로세스에 의해 사용자의 인터럽트 벡터가 소비되지 않으며 인터럽트 벡터를 변경하는 데는 프로그램의 코드영역을 변경할 필요가 있는 것에 반하여, 이 방법에서는 통신번호가 데이터로서 취급되고 있기 때문에 변경이 쉽다는 등의 장점이 있다.

이상에서 설명한 이외의 벡터(28H, 29H, 2BH ~2EH, 30H~3FH)는 앞으로의 확장을 위해 예약된 것이다. 일부의 기종에서는, 그들 벡터를 특정용도로 할당하고 있는 것도 있지만, 그들은 기종에 따라 차이가 크므로 여기서는 설명하지 않을 것이다.

펑션 콜

INT 21H에 의해 호출된 시스템 콜은, AH레지스터의 값에 따라서 여러가지 서비스를 프로세스에 제공한다. 그 내용은 분기하여 넘기는 것으로, 일반적인 프로그램에서는 이 INT 21H 시스템 콜만으로 모든 처리를 수행할 수 있다.

C언어에 의해 프로그램을 개발하는 경우에는 시스템 콜을 직접 이용하지 않더라도 처리계에 부속한 라이브러리 함수를 이용하는 것에 따라, 제닉스(XENIX) 및 그 외의 운영체계로의 보다 이식성이 높은 프로그램을 작성할 수 있다. 그러나, 디렉토리 액세스를 수행하는 프로그램 및 상주 프로세스가 된 프로그램과 같이 MS-DOS에 밀착한 프로그램을 기술하는 경우에는 유닉스(UNIX)와 호환적인 라이브러리 만으로는 불충분하여, 여하한 형식으로든 펑션 콜을 직접 기동할 필요가 있다.

(1) 펑션 콜의 분류

펑션 콜은 크게 나누어, CP/M 호환적인 것과, 제닉스(XENIX) 호환적인 것의 2개로 나눌 수 있다. 둘다, 서로 유사한 기능을 갖는 펑션 콜을 갖고 있기 때문에 프로그램을 작성하는 측에서 보면, 어떤 서비스를 받고 싶을 때, 대부분의 경우에는 2개의 펑션 콜 중에서 어느 것을 선택할 것인가 고민하게 된다. 따라서 펑션 콜을 이용하는 경우에는 둘의 설계개념과 기능의 차이를 이해하여, 각각의 프로그램에 어떤 펑션 콜을 선정할 것인가를 결정해야 하는 것이다.

(2) CP/M 호환

CP/M과 호환적인 부분은 초기의 MS-DOS에서 CP/M의 프로그램을 쉽게 이식할 수 있도록 준비한 것이다. 그 특징은 CP/M과 마찬가지로 키보드, 프린터 등의 각 디바이스를 각각 별개의 것으로 취급하고, 기능이 다른 서비스는 다른 펑션 콜로서 취급한다고 하는 것이다.

예를 들면 키보드로부터의 1문자 입력에 있어서 에코백(Echo Back)의 유무, 문자의 입력을 갖는지 어떤지 등의 차이에 따라, 몇개의 펑션 콜을 준비하고 있다. 또 디스크의 입출력도, CP/M에 준한 것이 준비되어 있어, 사용방법은 매우 복잡하게 된다.

(3) 제닉스호환

제닉스 호환 펑션 콜은, 버전 2.0이후의, 계층화 디렉토리를 비롯한 여러가지의 확장에 대응한 것과 이전의 MS-DOS상의 프로그램을 제닉스로 쉽게 이식하는 것을 목적으로 한 것이 준비되어 있다. 따라서 그 특징으로서는 계층화 디렉토리 및 화일 핸들(File Handle) 등, 제닉스와 같은 기능도 취급할 수 있다. 예를 들면, 어떤 디바이스 및 화일에 대해서도 같은 펑션 콜로 읽고 쓸 수 있고, 에코백을 하는지 어떤지라는 제어도, 보다 보편적으로 '디바이스 기능의 제어(I/O제어)'라는 펑션 44H인 펑션 콜에 의해 콘솔(Console)의 모드를 변경하고 나서 읽어낸다고 하는 수단을 취하고 있다. 또, 펑션 콜이 돌려주는 에러 값이 통일되어 있어 (어떤 화일에서도, 번호로부터 에러의 종류를 알아낼 수 있다), 그런 점에서도 쉽게 취급할 수 있게 되어 있다.

이와같은 설계개념의 차이는, 기능적으로 표 2와 같이 정리할 수 있다.

또, 하나의 프로그램에서 CP/M호환 콜과 제닉스 호환 콜의 어느쪽 하나 만을 사용할 수 있는 것이 아니라, 필요에 따라 사용할 수 있는 것이다. 또한 양자의 혼용에 의한 혼란을 피하고, 계층화 디렉토리 등의 기능을 이용하기 때문에, 대부분의 경우 제닉스 호환 콜을 이용하고 있다.

C언어와 어셈블리 언어의 링크

C언어에 의한 프로그램은 대부분의 경우 충분히 고속이고, 호환 콜의 호출도 라이브러리함수를 이용해서 무리없이 가능하므로, 어셈블리언어 루틴을 사용하지 않아도, 대체로 프로그램은 작성할 수 있다. 그러나, 특별히 고속성이 요구되는 프로그램 및 라이브러리함수에서는 준비되지 않은 펑션 콜, 특수한 시스템 콜을 이용하는 경우에는 C언어에 의한 기술만으로는 불충분하다. 예를 들면, 앞에서 설명한 INT 25/26H의 시스템 콜을 이용하면, 레지스터의 내용이 모두 파괴되어 버리지만 마이크로소프트C에서는 일부의 레지스터라도 내용을 파괴하면 안되기 때문에 이들의 시스템 콜은 C언어 프로그램에서 직접 기동하는 것은 불가능하고, 일단 어셈블리언어 루틴을 호출하여, 그 내부에서 레지스터의 내용

〔표 2〕 CP/M과 제닉스 호환콜의 비교

CP/M 호환	제닉스 호환
• 디바이스, 기능마다 다른 펑션이 준비되어 사용방법을 숙지해야 함. • MS-DOS기능의 일부를 지원하지 않는다. • 디바이스 액세스할 때 관리정보를 FCB에 의해 사용자가 관리해야 한다.	• 가능한 한 통일적인 펑션콜이 제공되고 있어, 디바이스의 기능에 관계없이 같은 펑션콜을 이용할 수 있다. • MS-DOS의 모든 기능을 지원한다. • FCB에 상당하는 정보를 MS-DOS가 관리하고 있기 때문에, 디스크 액세스가 쉽게 가능하다.

〔표 3〕 C와 어셈블리 프로그램 혼용시의 주의사항

• 루틴의 입구와 출구에서의 처리 • 인수의 액세스 방법 • 글로벌변수의 액세스방법 및 다른 함수의 호출방법 • 값을 돌려주는 방법 • 메모리모델에 따른 변동사항 • 작업용 영역

을 대피하고 나서 시스템 콜을 기동할 필요가 있다.

반대로, 어셈블리언어의 프로그램에서 C언어의 라이브러리를 포함한 함수를 호출하는 경우도 있을 수 있다. 예를 들면, 어떤 프로그램의 개발에 있어서 기본적으로는 속도때문에 어셈블리언어로 작성하지만, 일부의 입출력에 대해서는 복잡한 서식제어를 하기 위해 C언어의 printf 등의 함수를 이용하고자 하는 경우이다. 이와같은 경우에는, C언어의 프로그램과 어셈블리언어의 프로그램을 링크하게 된다.

여기서는, 이와같은 C언어와 어셈블리언어의 프로그램을 링크할 때 주의를 기울여야 할 사항을 **표 3**에 정리하였고 이하에 조금 구체적으로 설명하였으며 마지막으로 C언어의 프로그램에서 라이브러리함수를 사용하지 않고 직접 펑션 콜을 실행하는 함수인 pcint, pcintx, 그리고 pcintx를 이용하기 위해 필요한 segread함수를 작성하고자 한다.

(1) 루틴의 입구와 출구에서의 처리

먼저, 함수의 입구와 출구의 처리는 마이크로소프트 C가 출력하는 것과 같도록 해 두는 것이 일반적이다. (이것을 입력하는 것이 번거롭다고 여기는 사용자는 '-Fa' 옵션을 사용하여 템플레이트를 만들고, 그것에 수정을 가하는 방법을 취하여도 좋지만, 버그에 의해 올바른 코드가 출력되지 않는 경우도 있으므로 주의해야 한다). si, di, ds, ss의 각 레지스터는, 그 함수 내부에서 사용하는 것이며, 이들을 push하여 둘 필요가 있다. 그렇지않으면, 특별히 push할 필요는 없다. 또, 플래그 레지스터 중, 전송방향을 결정하기 위한 디렉션(Direction)플래그는 항상 제로(0)로 해두어야 하고, 1로 한 경우에는 함수가 종료할 때가지 cld명령을 사용하여 제로(0)로 해둔다. 이것은 마이크로소프트에서는 구조체 및 double의 값 등의 이동에 블록전송을 사용하고 있기 때문이다.

〔표 4〕 대문자와 소문자의 구별지시 예

매크로어셈블러 MASM	
• 대,소문자의 구별없이 작성한 프로그램의 어셈블	• 대,소문자의 구별하여 작성한 프로그램의 어셈블 소문자를
/MX 옵션 사용	/ML 옵션 사용
예) MASM 화일명 /MX ;	예) MASM 화일명 /ML, 실행형 화일명 ;
링커 LINK — /NOI 옵션 사용 예) LINK A.OBJ, B.OBJ, 실행형 화일명 /NOI ;	

(2) 함수 및 변수이름

C언어과 어셈블리 언어의 링크는, 일반적으로 심볼을 매개로 하여 수행되지만, 컴파일러, 어셈블러, 링커 등의 처리계에 의해 사용자가 지정한 심볼은 가공되고, 또 그 가공방법은 처리계의 버전마다 다르다.

C언어처리계에서의 심볼가동

마이크로소프트 C에서는, 프로그램에 쓰여진 심볼의 앞에 밑선(Under-Bar)을 붙여 오브젝트 화일(.OBJ)을 출력한다. 따라서 C언어의 프로그램에서 호출되는 어셈블리 언어의 프로시듀어 이름은, C언어의 프로그램에서 참조하고 있는 이름 앞에 밑선(_)을 붙이지 않으면, 링크할 때 에러가 발생한다. 예를 들어 C언어의 프로그램에서 'pcfunc'이라는 이름의 어셈블리 언어 프로시듀어를 호출하는 경우, 어셈블리 언어의 프로그램에서 해당하는 프로시듀어에 '_pcfunc'라고 이름을 붙여야만 한다.

반대로 어셈블리 언어 프로그램에서 C언어의 함수를 호출하는 경우에는, 함수이름 앞에 밑선을 붙인 형으로 참조하게 된다. 예를 들면, 라이브러리의 printf함수를 호출하는 경우에는 '_printf' 라는 심볼로 참조하게 된다.

C언어의 프로그램에서 참조되는 심볼이 모두 밑선으로 시작된다는 것을 이용하면, 어셈블리 언어로 기술하는 프로시듀어 및 변수 등의 심볼을 밑선 이외의 문자로 시작하게 하여, 그 심볼을 C언어의 프로그램에서 참조하지 않

는 것으로 할 수가 있다. 이와같은 이용방법은 그 심볼을 잘못하면 C언어에서 접근할 수 없기 때문에 버그가 발생하기 힘들게 되고, 프로그램을 읽는 경우에도 그 심볼이 어셈블리 언어의 프로그램에서만 참조하는지 어떤지를 확실히 알 수 있기 때문이다.

• 대문자와 소문자로 구별

MS-DOS의 매크로어셈블러(MASM) 와 링커(LINK)는 기본적으로 대문자와 소문자를 구별하지 않는다. 한편, 마이크로소프트 C에서는 이들을 구별하기 때문에 대문자와 소문자의 구별이 없는 심볼이면 올바른 결과를 얻을 수 없다.

그러나, 버전3.1 이후의 MASM과 LINK에서는 옵션에 의해 이들을 구별하도록 지시할 수 있게 되어 있어 심볼의 식별에 관한 문제는 해소되고 있다. 따라서, C언어에 의해 프로그램을 개발하는 경우에는, 버전3.1 이후에 MASM 과 LINK를 사용할 때는 반드시 옵션에 의해 대문자와 소문자의 구별을 하도록 지시할 필요가 있다. 표 4에 그 예를 보인다.

(2) 레지스터의 보존

C언어의 프로그램에서 어셈블리 언어의 프로그램을 기동할 경우에는, 일부 레지스터에 대해서는 그 값을 파괴해서는 안되는 것으로 되어 있다. 어셈블리 언어의 프로시듀어에서 그들의 레지스터 값을 변경하는 경우, 그들 레지스터를 일단 대피해 둘 필요가 있다. 마이크로소프트 C의 경우에는 일반적인 레지스터로서 SP, BP, SI, DI 레지스터와 세그먼트 레지스터인 CS, DS, SS레지스터, 그리고 항상 제로(0)인 디렉션 플랙 등의 레지스터 내용은 파괴하지 말하야 하는 것으로 되어 있다.

〔프로그램 1〕 어셈블리 언어에 의한 함수의 작성

```
**** C 언어 프로그램 ****

nothing(i, c)
int  i;
char c;
{
         int j;
         j = c;
}

**** 어셈블리 언어 프로그램 ****

_TEXT            segment byte public 'CODE'  ←— 마이크로소프트
                 assume cs: _TEXT                 C인 경우

stkframe         struct   ←—┐
         j       dw       ?  └— 스택프레임의 구조를 선언한다.
         bpsave  dw       ?
         retaddr dw       ?
         i       dw       ?
         c       dw       ?
stkframe         ends

                 public   _nothing
_nothing         proc     near
                 push     bp
                 mov      bp, sp
                 sub      sp, bpsave

                 mov      ax, [bp-bpsave].c
                 mov      [bp-bpsave].j, ax

                 mov      sp, bp
                 pop      bp
                 ret

_nothing         endp
_TEXT            ends
                 end
```

〔표 5〕 함수의 값을 돌리는 방법

값의 형식	데이터길이	값을 돌리는 레지스터
(unsigned) char	16비트	ax
(unsigned) short	16비트	ax
(unsigned) int	16비트	ax
(unsigned) long	32비트	dx(상위), ax(하위)
float	32비트	주소=ax
doubled	64비트	주소=ax
struct/union	—	주소=ax
near포인터	16비트	ax
far포인터	32비트	dx : ax

(4) 인수의 교환

C언어에서 함수의 인수는 스택에 쌓여 넘어가지만 그때 자동적으로 형 변환을 한다. 예를 들면, char형의 값은 int형으로 float형의 값은 double형으로 변환되고 나서 스택에 쌓이지만 이 규칙은 C언어와 어셈블리 언어를 링크하는 경우에도 마찬가지이다.

또 인수가 여러개인 경우, C언어의

〔프로그램 2〕 어셈블리언어에서 C함수의 호출방법

```
 ****  어셈블리 언어 프로그램  ****

 EXTRN    _printf:near  ←── 스몰모델인 경우에는 near사용

 CONST    segment
 $STR1    db       'hello %s',   00H  ←── 문자열은 이렇게 선언한다
          even
 CONST    ends

  TEXT    segment

          push     word ptr _p ←── 인수는 마지막 것부터 스택에
          lea      ax, $STR1           쌓인다.
          push     ax
          call     _printf ──────────────────┐
          add      sp, 4  ←─┐                │
                       인수는 호출한 쪽에서      │
  TEXT    ends             치워둔다.            │
─────────────────────────────────────────────│──
 ****  C 언어 프로그램  ****                     │
                                              │
          printf("hello %s", p);  ←───────────┘
```

기술차례와는 반대로, 최후의 인수부터 차례로 스택에 쌓여져 제1인수가 최후에 쌓이게 된다. 스택에 쌓여진 인수는 호출한 측에 의해 스택으로부터 꺼내어진다.

프로그램 1은 C언어 함수를 어셈블리 언어의 프로그램으로 치환한 것이지만 인수의 형 변환은 스트럭쳐 stkf-rame 과 같은 형식으로 수행되고 인수는 c, i의 순서로 스택에 쌓인다. 반대로, 어셈블리 언어로부터 C언어의 함수를 호출한 예를 **프로그램** 2에 보인다.

(5) 함수 값을 되돌리는 방법

보통 C언어의 함수는 파괴해도 좋은 레지스터에, 돌려주는 값을 격납하여 호출한 쪽에 돌려준다. 마이크로소프트 C의 경우도 예외가 없고, 돌려주는 값의 형과 값이 격납되는 레지스터는 표 5와 같다.

함수의 값을 돌려주는 데는 표 4의 규칙에 따르도록 한다. 우선 8내지 16비트의 값은 ax를 사용하여 돌려주고, 32비트의 값은 dx(상위)와 ax(하위) 레지스터를 사용하는 것이 기본이다. struct 혹은 union의 경우는 메모리 안에 결과를 격납해 두고, 그 선두주소를 ax에 세트(Set)하여 돌려준다. 라지 모델인 경우에는 세그먼트 주소를 dx에 넣어 돌려준다. float 및 double형의 경우도 마찬가지로, 메모리 안에 결과를 격납하여 그 선두주소를 ax에 세트하여 돌려주고 라지 모델인 경우에는 세그먼트 주소를 dx에 넣어 돌려준다. 마이크로소프트 C에서는 float 및 double 의 값을 돌려주기 위해 '_fac'라는 8바이트의 영역을 사용하고 있으므로 어셈블리 루틴에서도 이 영역을 사용하면 좋을 것이다. 또, near 포인터의 경우에는 이것을 ax에 세트하여 돌려주지만 far 포인터의 경우에는 dx레지스터를 사용하여 세그먼트 주소를 돌려준다. PC

전화번호부 프로그램(I)

박성현 /디베이스 프로그래머

프로그램을 배우는데 있어서 가장 효과적이고 빨리 배우는 방법은 훌륭한 프로그램의 소스를 읽고 이해하고 이를 토대로 더욱 훌륭한 프로그램으로 만드는 과정에서 얻을 수 있다고 여러 유명한 프로그래머들은 말했다. 필자 역시 이러한 과정을 거쳐 프로그래머가 되었으며, 이 강좌 역시 프로그램의 기초를 설명한 후 일상 생활에 유용하게 쓰일 수 있는 전화번호부 프로그램의 제작과정을 3회에 걸쳐 소개하고자 한다.

우리가 흔히 필요로 하는 각 개인의 전화번호와 주소를 관리하고자 하는 경우 주소, 전화번호를 입력하여 수시로 검색하여 알아보고 이러한 데이터를 이용하여 주소록을 만들거나 우편용 레이블을 만들 수도 있을 것이다.

이 프로그램 리스트는 알아보기 편하도록 IF, CASE, LOOP문에 줄을 그어 표시하였으며, 설명의 편리를 위하여 앞에 행번호를 붙였다. 독자가 이 프로그램을 코딩할 때는 문번호와 라인을 삭제하고 타자하여야 할 것이다.

디베이스 자체의 내장 워드프로세서를 사용할 경우 5천 바이트가 넘게 되면 뒷부분의 프로그램을 잃어버리게 되므로 반드시 독자가 사용하는 전용의 에디터나 워드프로세서를 사용하길 바란다. 특히 삼보 컴퓨터에서 제공되는 TGEDIT 등의 일부 에디터는 디스크 공간을 아끼기 위하여 저장할 때 무조건 SPACE를 탭으로 바꾸어 저장하는 것이 있으므로 TGEDIT의 경우는 저장할 때 'save / n'이라고 반드시 옵션을 주면서 저장해야 한다. 그외의 워드프로세서 등에서도 화면의 폭이 좁아졌을 경우 다시한번 사용법을 익히고 해보길 바란다.

MAIN.prg의 34번 행의 경우의 예를 들어 보자.

```
@ 7,22 SAY '(1) 입              력'
```

맨앞의 8칸은 탭이든 SPACE든 상관없이 수행이 되나 "(1)"과 "입" 그리고 "력" 사이의 공간이 SPACE가 탭으로 변했다면 디베이스에서는 화면 출력을 '(1)○○입○○○력'으로 나오게 된다. 워드프로세서나 프린터의 경우 탭은 1바이트의 캐릭터가 나오면 화면이나 프린터에서 지정한 만큼 띄어주나 디베이스에서는 탭이라는 캐릭터가 나오면 탭 마크를 표시해주고 화면의 폭을 띄워주지 않는다. 그러므로 반드시 화면에 디스플레이되는 부분은 탭을 사용하지 말고 SPACE를 사용하여야 하고 무조건 탭으로 바꾸면서 저장하는 에디터의 경우는 주의하여 사용하여야 한다.

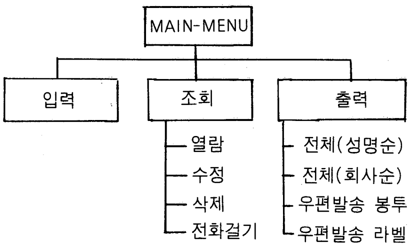
전화번호부 프로그램의 구조는 그림 1과 같으며, 프로그램의 리스트는 MAIN.prg, INPUT.prg, SCREEN.prg, OUTPUT. prg, READ.prg로 구성된다.

전화번호부 프로그램을 코딩하기 전에 우선 다음과 같은 데이터베이스 화일을 만들기 바란다.

데이터베이스 화일 만들기

데이터베이스 화일을 만드는 명령은

〔그림 1〕 전화번호부 프로그램의 구조



CREATE이고 그 화일의 구조를 변경하는 명령은 MODIFY STRUCTURE임을 다시한번 기억하고 먼저 전화번호부를 만들기 위해 필요한 필드를 생각해 보자.

성명: 3자 또는 4자의 이름이 필요하다. 한글은 1자당 2바이트가 필요하므로 8자까지 필요하겠으나 여유를 두기 위해 10자로 지정한다.

```
NAME  Character 10
```

주소: 20자 정도면 충분하다. 더 길면 줄여서 기록할 수 있다.

```
ADDR  Character 40
```

전화번호: 국제 전화번호까지 대비해서 15자리까지 2개를 정한다.

```
TEL1  Character 15
TEL2  Character 15
```

회사명: 20자 정도로 여유를 두어 지정한다.

```
COMP  Character 40
```

직위:

```
JIK   Character 10
```

우편번호 :

```
WOO   Character 7
```

위의 데이터 구조를 CREATE해서 새로운 데이터화일을 만들고

```
USE CARD
LIST STRUCTURE
```

를 입력하면 다음과 같은 CARD.DBF의 구조가 나타난다.

```
Structure for database: C:CARD.dbf
Number of data records:      10
Date of last update   : 89.04.05
Field  Field Name  Type        Width    Dec
    1  NAME        Character      10
    2  COMP        Character      40
    3  JIK         Character      10
    4  ADDR        Character      40
    5  WOO         Character       7
    6  TEL1        Character      15
    7  TEL2        Character      15
** Total **                      138
```

인덱스화일 만들기

다음은 이 데이터화일이 어떤 순서로 위치하고 있어야 가장 쉬운가 생각해보자. CODE는 필요없으니 성명 순으로 위치시키는 것이 가장 편리하다. 하나를 더 만들어 보면 회사이름 순으로 위치시켜 다음과 같이 회사이름으로도 데이터를 체크할 수 있도록 한다.

```
USE CARD
INDEX ON NAME TO CNAME
INDEX ON COMP TO CCOMP
```

MAIN.prg의 분석

리스트 1은 MAIN.prg이며, 다음은 각 명령라인에 대한 설명이다.

1-5 : 프로그램의 설명을 위한 코멘트부분이다. 디베이스에서는 명령어 라인의 처음에 '*' 마크가 있으면 실행하지 않는 설명을 위한 라인으로 간주한다. 이곳에서는 프로그램의 이름과 작성자, 작성일과 프로그램의 중요한 내용을 적어 놓아 차후에 어떠한 프로그램인지 쉽게 알 수 있도록 한다. 이 프로그램에서는 설명을 이 부분에서 따로 하기 위하여 프로그램 내에서는 코멘트를 최소화하였지만 프로그램소스에는 코멘트를 많이 적어 놓을수록 차후에 관리하고 수정하기에 용이함은 아무리 강조하여도 지나치지 않을 것이다.

6 : 한줄이 띄어 있어도 프로그램 진행에 아무 영향을 미치지 않으므로 프로그램이 한눈에 보이도록 한줄씩 띄워 소스를 읽기 쉽도록 할 수 있다.

7-25 : 메인메뉴가 나오기전에 프로그램 전체의 환경을 설정해준다.

7 : 비 대화식모드로 설정하여 프로그램 실행시 응답이 나오지 않도록 한다.

8 : 프로그램 명령어라인이 보이지 않도록 한다. 디폴트가 OFF이므로 여기서 지정해주지 않아도 되지만 프로그램 실행전에 SET ECHO ON 상태였다면 프로그램 실행시 앞서 OFF로 바꿔주기 위함이다.

9 : 20번째 라인에 나오는 STATUS BAR(현재 열려있는 화일과 Caps Lock, Num Lock과 Ins 키 상태 등을 표시해주는 부분)를 없애준다.

10 : STAUS BAR가 없을 경우는 Caps Lock, Num Lock과 Ins 키 상태가 화면의 0번 라인에 나오게 되는데 이것도 없애준다.

11 : 디베이스 프로그램실행 중에 Esc 키를 누르면 프로그램을 중지하게 되는데 실수로 Esc키를 눌렀을 경우를 대비하여 Esc키로 인한 인터럽트를 할 수 없도록 해준다.

12 : 필드를 꽉차게 입력하였거나 에러 발생시 벨소리가 나오는 것을 나오지 않도록 해준다.

13 : 디베이스에서는 레코드를 하나 삭제할 경우 실제로 삭제를 하지 않고 레코드의 맨앞에 '*'표를 추가해서 삭제를 암시하고 레코드를 다시 재생하기 위해서는 RECALL이라는 명령을 사용한다. 실제로 데이터를 삭제하였더라도 삭제된 레코드는 계속해서 보이는데 삭제된 레코드 -앞에 '*'가 있는 레코드-는 없는 것처럼 간주하도록 하는 환경설정이 바로 SET DELETED ON이다. 이 환경설정하에서는 삭제마크 표시된 레코드는 보이지 않게 된다. 디베이스 명령중 삭제마크된 레코드를 완전히 없애주는 명령으로 PACK이라는 명령이 있으나 레코드의 갯수에 따라 많은 시간을 요하는 명령이므로 사용을 피해야 한다. 특히 프로그램 내에서는 이 명령을 절대 사용해서는 안되고 SET DELETED ON 상태에서의 DELETE라는 명령으로만 삭제를 해서 프로그램의 속도에 효율을 기해야 한다.

14 : 디폴트로는 'mm / dd / yy' 형태로 나오나 dBASE Ⅲ+에서는 여러국가에서 사용할 수 있는 형식을 갖추고 있다.
이 명령을 사용하면 'yy.mm.dd' 형태의 날짜로 보여준다.

15 : GET을 사용하면 디폴트로 반전의 색으로 구분이 되나 반전의 색을 없애면 아무 구분이 없어진다. 이때 '[]'로 구분이 되도록 하기 위한 명령이다. 여기서 다른 기호를 사용할 수도 있다.

16 : SET DELIMITER TO 〈expC〉의 명령이 실제 작동이 되게 하는 명령이다.
디폴트로 작동되게 하기 위해서는 SET DELIMITER OFF를 하면 된다.

17 : 오늘 날짜를 기억시켜 둔다. 프린트시 나오게 될 날짜를 기억해둔다. 메인메뉴에서 'D'를 입력하여 날짜를 고칠 수 있게 하였으므로 메인메뉴에 들어가기 전에 한번만 시스템 날짜를 기억해둔다.

〔리스트 1〕 MAIN.PRG

```
* Program.: MAIN.PRG
* Author..: Bahk Seonghyeon.
* Date....: 1989. 4. 5.
* Notice..: 전화 번호부 메인메뉴
* Notes...: Called from  SYSTEM

SET TALK       OFF
SET ECHO       OFF
SET STATUS     OFF
SET SCOREBOARD OFF
SET ESCAPE     OFF
SET BELL       OFF
SET DELETED    ON
SET DATE       ANSI
SET DELIMITER  TO '[]'
SET DELIMITER  ON
today=DATE()
IF FILE('CNAME.NDX') .AND. FILE('CCOMP.NDX')
¦    USE CARD INDE CNAME,CCOMP
ELSE
¦    USE CARD
¦    INDEX ON NAME TO CNAME
¦    INDEX ON COMP TO CCOMP
¦    SET INDEX TO CNAME,CCOMP
ENDIF

DO WHILE .T.
¦    SET COLOR   TO
¦    CLEAR
¦    SET CENTURY ON
¦    @  3,6  SAY REPLICATE([-],69)
¦    @  5,6  SAY REPLICATE([-],69)
¦    @  4,7  SAY REPLICATE([≣],7)+[ 전 화 번 호 부 ];
¦                +REPLICATE([≣],7)
¦    @ 18,6  SAY REPLICATE([-],69)
¦    @  7,22 SAY '(1)            입                 력'
¦    @ 10,22 SAY '(2)            조                 회'
¦    @ 13,22 SAY '(3)            출                 력'
¦    @ 16,22 SAY '(Q)            종                 료'
¦    @ 19,24 SAY REPLICATE([-],31)
¦    @ 21,24 SAY REPLICATE([-],31)
¦    @ 19,11 SAY [D A T E]
¦    @ 19,62 SAY [T I M E]
¦    @ 22,6  SAY REPLICATE([-],69)
¦    SET COLOR   TO N/W
¦    @ 20,10 SAY today
¦    @ 20,60 SAY [00:00:00 AM]
¦    SET COLOR   TO W/N+
¦    @ 20,27 SAY '당  신  의  선  택  은  [?]'
¦
¦    cho = 0
¦    DO WHILE cho = 0
¦    ¦    SET COLOR TO N/W
¦    ¦    time=TIME()
¦    ¦    ntime=VAL(SUBSTR(time,1,2))
¦    ¦    DO CASE
¦    ¦    ¦    CASE ntime > 12
¦    ¦    ¦    ¦    time=RIGHT(STR(ntime-12)+SUBSTR(time,3,6)
¦    ¦    ¦    ¦         +[ P],10)
¦    ¦    ¦    CASE ntime = 12
¦    ¦    ¦    ¦    time=time+[ P]
¦    ¦    ¦    CASE ntime < 12 .AND. ntime > 00
¦    ¦    ¦    ¦    time=time+[ A]
¦    ¦    ¦    CASE ntime = 00
¦    ¦    ¦    ¦    time=[12]+SUBSTR(time;3,6)+[ A]
¦    ¦    ENDCASE
¦    ¦    @ 20,60 SAY time
¦    ¦    cho = INKEY()
¦    ¦    @ 20,52 SAY []
¦    ¦    SET COLOR TO W/N+
¦    ¦    IF cho < 0
¦    ¦    ¦    cho = 0
¦    ¦    ¦    LOOP
¦    ¦    ENDIF
¦    ¦    IF UPPER(CHR(cho))$[123Q]
¦    ¦    ¦    EXIT
¦    ¦    ENDIF
¦    ¦    IF UPPER(CHR(cho))$[D]
¦    ¦    ¦    SET DELIMITER OFF
¦    ¦    ¦    @ 20,10 GET today
¦    ¦    ¦    READ
¦    ¦    ¦    SET DELIMITER ON
¦    ¦    ENDIF
¦    ¦    cho = 0
¦    ENDDO
¦
¦    cho=UPPER(CHR(cho))
¦    @ 20,52 SAY cho
¦    SET COLOR TO
¦    SET CENTURY OFF
¦
¦    *-----------------------------------------------------*
¦
¦    DO CASE
¦    ¦    CASE cho=[Q]
¦    ¦    ¦    CLOSE ALL
¦    ¦    ¦    SET TALK       ON
¦    ¦    ¦    SET ESCAPE     ON
¦    ¦    ¦    SET SCOREBOARD ON
¦    ¦    ¦    SET STATUS     ON
¦    ¦    ¦    SET DELIMITER  OFF
¦    ¦    ¦    SET BELL       ON
¦    ¦    ¦    SET DELETED    OFF
```

```
102 ¦   ¦   ¦   RETURN
103 ¦   ¦  CASE cho=[1]
104 ¦   ¦   ¦   DO INPUT
105 ¦   ¦  CASE cho=[2]
106 ¦   ¦   ¦   DO READ
107 ¦   ¦  CASE cho=[3]
108 ¦   ¦   ¦   DO OUTPUT
109 ¦   ENDCASE
110 ENDDO
111 * End-of-file MAIN.prg
```

18-25 : 데이터베이스 화일을 연다. 만약 인덱스화일이 있다면 인덱스화일과 함께 같이 열고 인덱스화일이 없다면 데이터베이스 화일만 열고 인덱스화일을 새로 만들고 인덱스화일 2개를 열어 놓는다.

18 : 만약 CNAME.NDX와 CCOMP.NDX라는 화일이 둘다 존재한다면

19 : CARD.dbf라는 데이터베이스 화일과 인덱스화일로는 CNAME.ndx와 CCOMP.ndx만 순서대로 열어 놓고 다른 화일은 모두 닫는다.

20 : 위의 IF조건에 해당되지 않는다면 실행한다.

21 : CARD.dbf라는 데이터베이스 화일을 열고 다른 화일은 모두 닫는다.

22 : 현재 열려져 있는 데이터베이스 화일의 필드중 이름 순서대로 목차를 가지고 있을 인덱스화일을 새로 만든다. 화일이름은 CNAME.NDX로 한다.

23 : 현재 열려져 있는 데이터베이스 화일의 필드중 회사명 순서대로 목차를 가지고 있는 인덱스화일을 새로 만든다. 화일이름은 CCOMP.NDX로 한다.

24 : 현재 열려 있는 데이터베이스 화일과 함께 CNAME.NDX, CCOMP.NDX 2개를 차례로 함께 열어 놓는다. 22번 라인을 실행한 후에는 인덱스화일이 CNAME.NDX 하나만 열려 있는 상태이고 23번 라인을 실행했을 경우는 CNAME.ndx는 닫아지고 CCOMP. ndx 하나만 열려있는 상태가 된다.

인덱스화일을 동시에 2개 열기 위해서 이 명령어를 사용한다.

25 : 가장 최근에 시작한 IF문을 끝낸다.

27-110 : 서브프로그램의 수행 후 다시 메뉴가 나오도록 하는 루프

28 : 화면의 컬러를 디폴트인 검은바탕의 흰글씨로 해준다. 반전되는 부분은 흰바탕에 검은 글씨로 한다.

29 : 화면의 전체를 지운다.

30 : [89.05.15]를 [1989.05.15]로 [05/15/89]은 [05/15/1989]로 연도를 4자리로 보여주도록 해준다.

31-48 : 메뉴화면의 내용을 디스플레이한다. (31, 32, 34, 39, 40, 44) REPLICATE([-], 69) : '-' 69개를 연결해서 화면에 나오도록 한다. 만약 독자의 컴퓨터의 기종이 7비트 완성형한글(청계천 한글카드)을 사용한다면 이 명령은 43행과 함께 @ 5, 6 TO 22, 75의 간단한 명령으로 박스를 그릴 수 있다.

33 : 만약 청계천 한글카드를 사용한다면 세미 그래픽이 1바이트로 가능하므로 REPLICATE(CHR(176), 14)로 하여야 할 것이다.

44-46 : 컬러를 반전시키고 날짜와 시간을 디스플레이한다.

47 : 밝은 글씨가 나오도록 한다. 화면의 컬러를 지정할 때 '+'를 사용하면 밝은 강조색상이 되고 '*'를 추가하면 점멸하게 된다.

50 : cho라는 변수에 숫자 0를 넣는다.

51-83 : 66행에 의해 원하는 입력이 받아지지 않는 동안 계속 수행하면서 현재의 시간을 계속 디스플레이한다.

52 : 컬러를 반전시킨다.(흰바탕에 검은 글씨로)

53 : 현재의 시각을 읽어 들인다.

54 : 시, 분, 초 중의 시간만을 정수로 바꾸어 기억한다.

SUBSTR(expC, n, m)—어느 문자스트링의 n번째에서 m개를 떼어서 일부만 가져오는 함수이다.

VAL()—Character인 수를 Numeric인 수로 바꾸는 함수

55-64 : AM, PM을 구분하기 위한 선택을 한다.(오후, 정오, 오전, 자정)

56 : ntime(54행에 의해 시간만을 정수로 가지고 있는 변수이다.)이 12보다 클 경우. 즉, 오후인 경우.

57 : ntime이란 변수는 현재 12보다 크므로 12를 뺌으로써 오후 몇시라는 시간이 된다. 이것은 숫자이므로 캐릭터로 바꾸는 함수 STR을 사용하고 다음에 현재 시간의 분과 초를 연결시킨다. 그리고 뒤에는 오후라는 기호로 한칸의 공백과 함께 'P'자를 연결시킨다. RIGHT(expC, n)는 스트링의 오른쪽에서 왼쪽으로 n개 만큼의 문자를 받아들이는 함수이다.

65 : 바꾸어진 시간을 디스플레이한다.

66 : INKEY()는 키보드로부터 입력을 받아들이는 명령이지만 입력하기를 기다리지 않고 입력이 없을 경우는 그값을 0으로 기억하고 입력이 있으면 아스키 코드의 정수값으로 기억한다.

67 : 커서가 [?]의 위치에 잠시 지나가게 하여 커서가 여기서 대기하고 있는 것처럼 보이게 한다.

69-72 : 아스키 코드가 0보다 작은 수가 입력되는 경우 0으로 재조정하게 한다.(FUNCTION KEY를 누를 경우 0보다 작은 값이 나온다.)

72 : 여기서 LOOP명령을 사용하지 않아도 자동으로 LOOP가 되지만 64행에서의 CHR()함수에 에러를 방지

하기 위해(64행을 거치지 않게 하기 위해) LOOP명령을 사용한다.

73-75: 원하는 값이 나온 경우 LOOP에서 빠져 나오게 한다. UPPER()은 소문자를 대문자로 바꾸는 함수.

76-81: 'D'를 누를 경우 날짜의 입력을 하게 한다.

77: 15, 16행에 의해 지정되고 지정된 것이 나오도록 되었지만 날짜 변경시 반전상태인 디폴트방식으로 GET-READ에 의한 입력이 되도록 한다.

78-79: 날짜를 수정한다.

82: cho변수를 0로 다시 지정해 줌으로써 '123Q'를 누르지 않고 다른 키를 눌렀을 경우 메뉴가 계속 나오게 한다.

85: cho변수를 아스키코드에 해당되는 캐릭터로 바꾸고 이를 다시 대문자로 바꾼다.

86: 입력한 값이 '?'부분에 나타나도록 해준다. INKEY() 함수에 의해 입력될 경우 키보드로 누른 글자가 화면에 나오지 않기 때문에 지정된 장소에 누르는 것처럼 보이게 하기위해 67행에 의해 커서가 계속 깜박거리게 하였으며 정확히 입력된 값에 한해 다시 디스플레이되게 한다.

87-88: 화면의 컬러를 디폴트로 바꾸어 주고 연도 나오는 방식도 다시 지정해준다.

91-109: 선택한 값에 따른 각각의 명령을 해준다.

93: 선택한 변수가 [Q]인 경우

94-101: 모든 화일을 닫고 지정했던 SET 환경을 되돌린다.

102: 프로그램의 수행을 중지하고 복귀한다.

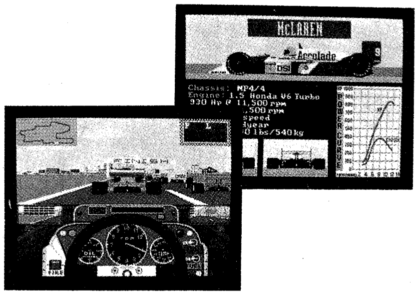
104: 이 프로그램은 여기서 잠시 멈추고 INPUT.prg라는 프로그램화일의 명령을 계속하라.

111: 프로그램의 끝임을 빨리 확인할 수 있도록 그리고 프로그램의 끝이 잘려서 완전한 프로그램인지 아닌지의 여부를 빨리 알 수 있도록 화일의 끝임을 표시해 주는 것이 좋다. PC

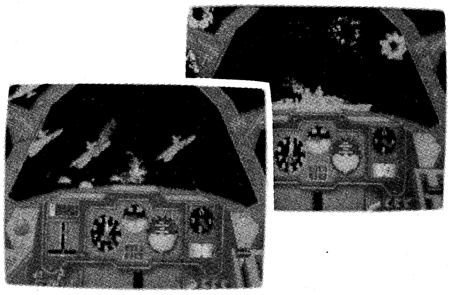
JACK NICKLAUS GOLF



GRAND PRIX CIRCUIT



DIVE BOMBER



10TH FRAME PROBOWLING	1
221B BAKER STREET	2
3D HELICOPTER	1
4TH & INCHES	1
ACE OF ACE	1
ALTERNATE REALITY	2
ANCIENT ART OF WAR AT SEA	1
BAD STREET BRAWER	1
BADUK	1
BALANCE OF POWER	3
BARD'S TALE I	2
BATTLE CHESS	2
BATTLE ZONE	1
BATTLEHAWKS 1942	2
BIONIC COMMANDO	1
BLACK CAULDRON	2
BLACK JACK ACAEMY	1
BLOCK BUSTER	1
BOP,N WRESTLE	1
BORROWED TIME	1
BUBBLE GHOST	1
CALIFONIA GAME	1
CARD SHARKS	1
CHAMPIONSHIP BASEBALL	1
CHAMPIONSHIP BOXING	1
CHAMPIONSHIP GOLF	1
CHUCK YEAGER'S FLIGHT TRAINNER	1
COMMANDO	1
CONTRA	1
CRAZY CARS	1
CRIME & PUNISHMENT	1
DAM BUSTER	1
DARK·CASTLE	2
DEATH SWORD	1
DEFENDER OF THE CROWN	1
DESTROYER	1
DOUBLE DRAGON	2
DRAGON WORLD	2
EARL WEAVER BASE BALL	2
EMPIRE	1
F-15 STRIKE EAGLE	1
F-19 STEALTH FIGHTER	3
FALCON F-16	1
FIRE POWER	2
FLIGHT SIMULATOR	1
FLIGHT SIMULATOR V3.0	2
FORD SIMULATOR	1
FRIENDLY WARE GAME	1
GAMESTAR BASEBALL	1
GATO	1
GAUNTLET	1
GHOSTS'N GOBLINS	1
GRAND PRIX CIRCUIT	2
GUNSHIP	2

HACKER II	1
HARD BALL	1
HARRIER	2
HEROES OF LANCE	3
HULK	1
HUNT FOR RED OCTOBER	1
IKARI WARRIOR	1
IMAGE TENNIS	1
IMPOSSIBLE MISSION II	1
INDOOR SPORTS	2
INFILTLATOR	1
INTO THE EAGLE'S NEST	1
JACK NIKLAUS GOLF	2
JEOPARDY	1
JET	1
KARNOV	2
KING'S QUEST I	1
KING'S QUEST II	2
KING'S QUEST III	3
KING'S QUEST IV	9
KNIGHT ORC	2
LAST NINJA	2
LEISURE SUIT LARRY II	6
LOCK ON	2
LODE RUNNER	1
MAD SPY VS SPY III	1
MAN HUNTER	5
MARBLE MADNESS	2
MASTER NINJA	1
MIGHT & MAGIC	2
MINI PUTT	1
MIXED UP MOTHER GOOSE	2
MOEBIUS	2
NFL CHALLENGE	2
NINJA	1
OFF SHORE WARRIOR	1
ONE ON ONE	1
PATTON VS ROMMEL	1
PETE ROSE PENANT FEVER	2
PHANTASIE	1
PHANTASIE III	1
PHYCHO	2
PINBALL WIZARD	1
PIRATES	2
PLATOON	2
POLICE QUEST	3
POLICE QUEST II	6
POOL OF RADIANCE	3
PT-109	1
QUSTRON II	2
RAD WARRIOR	1
RAMPAGE	1
RINGS OF ZILFIN	2
ROADWAR 2000	1

ROMANCE OF THREE KINGDOM	3
SCRABBLE	1
SDI	1
SEARCH AND DISTROY	1
SEVEN CITY OF GOLD	1
SHADOW GATE	2
SHANGHAI	1
SHOGUN	1
SILENT SERVICE	1
SKATE OR DIE	2
SKY FOX II	1
SKY RUNNER	1
SPACE QUEST I	2
SPACE QUEST II	3
STAR COMMAND	3
STAR FLEET·I	1
STAR FLIGHT	2
STAR GLIDER	1
STAR TREK	1
STREET FIGHTER	2
SUB BATTLE	1
SUMMER GAME II	1
SUPERBYKE	1
SUPERSTAR ICEHOCKEY	1
TEST DRIVE	1
TETRIS	1
THE GAMES SUMMER GAME EDITION	4
THE LAND OF THE LOUNGE LIZARDS	2
THEXDER	2
THREE STOOGES REEL	2
TOP GUN	1
TREASURE ISLAND	2
TROJAN	1
TWILIGHT RANSOM	2
ULTIMA I	1
ULTIMA II	3
ULTIMA III	2
ULTIMA IV	2
ULTIMA V	4
VIDEO VEGAS	1
WHERE IN THE EUROPE IS CARMEN-	1
WHERE IN THE USA IS CARMENSAN	1
WHERE IN THE WOLRD IS CARMEN	1
WILLOW	2
WINTER GAMES	1
WIZARD WARS	2
WIZARDRY IV	1
WIZARDRY I	2
WIZARDRY III	1
DIVE BOMBER	1
WORLD GAME	1
WORLD KARATE CHAMPIONSHIP	1
WORLD TOUR GOLF	1

마우스 드라이버 툴키트

宋家네/연세대 전자과

매킨토시나 PS/2용 소프트웨어들은 100% 마우스를 지원하는데 비해 IBM-PC의 경우 약 2만여종의 소프트웨어중 7~10%만이 마우스를 지원하고 있는 실정이다. 그러나 최근들어 마우스에 대한 관심도가 높아지고 있다. 이러한 마우스는 어떻게 구성되며 구성원리는 어떠한 가를 예제 프로그램 등을 통해 상세히 설명한다.

마우스란?

마우스란 누구나 알듯이 "쥐"란 뜻을 가진 단어다. 그런데 컴퓨터의 세계에선 이 마우스가 2가지 의미로 통용된다. 하나는 '마이크로 마우스'라고 해서 흔히 미로를 찾는 로보트 쥐를 의미하고, 또 다른 뜻으로는 이제 논의하게 될 입력 및 지시 장치인 '타블렛 마우스'이다.

마우스는 키보드와는 독립적으로 동작하는 새로운 스타일의 입력장치이다. 컴퓨터의 뒷쪽을 보면, 키보드와는 달리 RS-232C라는 표준 직렬 인터페이스카드에 쥐꼬리(연결코드)가 접속되어 있음을 볼 수 있다. 마우스의 모든 동작보고는 이러한 직렬 전송선과 일종의 통신으로 이루어지기 때문에, 만약 키보드가 삑삑거리며 죽었을지라도 마우스의 커서는 정상적으로 동작한다.

통신이라는 것은 상호간의 약속으로 이루어지는데, 이 약속을 프로토콜이라고 부른다. 우리가 사용하는 대부분의 마우스는 앞에서 언급한 것처럼 RS-232C라는 직렬 인퍼페이스카드를 이용하므로 직렬포트에 있는 UART(Universal Asynchronous Receiver / Transmitter)라는 IC칩을 이용해야 가능하다.

이 UART(=ACE, ACIA)칩은 비동기 직렬통신에 있어서 데이터를 쪼개거나 합쳐서 주고 받는 역할을 해준다. 따라서 마우스도 여러 정보(좌표 변위, 버튼 상태)를 UART 칩을 통해서 데이터의 규격을 알려주고, 램에 있는 마우스 드라이버로 하여금 모아진 데이터 형식에 대한 프로토콜을 표 1과 같이 갖게 한다.

이 부분에 대한 이해는 만약 다른 디바이스(타블렛이나 유저가 개발한 장치)로 마우스 드라이버를 이용할 경우에도 필요하다. 즉 기본적인 정보의 직렬분리와 원래의 1바이트로의 조합은 UART 칩에서 해주지만, 정보를 5바이트씩 읽고 해석해 내는 것은 각종 마우스 드라이버가 해낸다.

먼저 포트로부터 들어온 바이트1의 상위 5비트가 10000이면 변화가 있음을 알고 다음의 4바이트를 읽는다. 바이트1의 나머지 3비트는 각각 버튼에 1:1 대응이 되어 만약 사용자가 버튼을 누르면 해당 비트가 0에서 1로 토글된다. (손을 버튼에서 뗄때까지) 따라서 마우스는 지금 현재의 버튼상태를 알 수 있으며, 뒤에 잇달아 들어오는 2바이트를 통해 최근에 기록된 구좌표에 대한 상대적 X, Y축 변위를 계산해 커서를 움직이게 되는 것이다.

만약 그 2바이트가 전송될 때 마우스가 또 움직였다면, 바로 다음에 오는 바이트4와 5의 값이 바뀌어 전송되므로 정확하게 위치가 산출된다. 마우스는 오른쪽 위방향으로 갈수록 X, Y 값이 증가하나, 화면 좌표는 그렇지 않다. 하지만, 이 모든 계산은 마우스 드라이버가 램에서 다 처리해 줌으로, 우리는 먼저 msmouse.sys와 같은 시스템 화일을 부팅시에 등록하거나 msmouse.com 등의 램상주 화일을 먼저 띄우기만 하면, 그 화일들이 INT 33H를 이용하여 앞으로 소개할 함수들을 구동시켜 주게 된다.

마우스의 친척들로는 타블렛(Tablet), 터치 스크린(Touch Screen), 트랙볼(Track Ball) 라이트 펜(Light Pen) 등이

〔표 1〕

RS-232C UART 세팅		마우스 프로토콜
보오율(Baud Rate)	: 1200	바이트 1 : 동기 바이트(10000+MRL)
데이터 비트	: 8	바이트 2 : 구좌표 X와의 상대적 거리 변위
스타트 비트	: 1	바이트 3 : 구좌표 Y와의 상대적 거리 변위
스톱 비트	: 1	바이트 4 : 바이트 2와 같으나 차이가 날 수 있음
패리티(Parity)	: 없음	바이트 5 : 바이트 3과 같으나 차이가 날 수 있음

있다. 이들 장치들은 키보드가 못하는 능력(자유자재로 이동, 곡선 처리, 신속성과 정확성…)들을 보완해 주는데, 해상도와 편리도 정도에 따라 선택된다. 말하자면 주로 1인치당 몇 포인트를 감지할 수 있나로(DPI : Dots Per Inch) 해상도를 따질때 마우스는 타블렛에 버금갈 정도로 성능이 뛰어나다(150~400DPI, 주로 200DPI).

본 글은 마우스의 구조에 대한 것은 아니나, 상식적인 측면에서 몇가지만 더 이야기하겠다. 우선 마우스는, 고해상도 타블렛이 지시하는 펜이나 포인터에서 발하는 자기력으로 여러 정보를 감지하는 방법과는 달리 새로운 구조를 갖고 있다.

마우스는 구조적으로 크게 볼타입(Ball Type)과 광학 센서타입(Optical Type)으로 나누고, 그 각각은 눈(버튼)의의 갯수 즉 눈이 1개냐 2개냐 3개냐로 다시 나눌 수 있다.

볼 마우스는 배꼽(마우스 밑면)에는 둥근 공이 있어서, 사.용자가 밀거나 이동시킬때 그 공이 뒹구는 방향과 정도를 감지함으로서 위치를 산출한다. 센서 마우스는 가로세로로 조밀하게 줄이 그어져 있는 조그만 깔판위에서 이동하며, 마우스의 배꼽에서 발산되는 빛이 되돌아 올때 지나친 X, Y축 교차점의 갯수와 방향을 알아내어 위상을 정한다(사진 1 참조).

볼 마우스의 하이테크는 뒹구는 볼에 따라 감지되어 지는 인코더의 정밀도에 있으며, 센서 마우스는 줄쳐있는 패드(깔판)의 조밀성 못지않게 렌즈를 통해 반사되어 나오는 광선에 대한 정확한 센서의 감지능력에 있다. 하지만 어느 것이나, 고유의 커스텀 IC를 사용하기 때문에 사용자가 임의로 조정할 수 없다.

일반적으로 센서방식의 쥐들이 더 비싸고 정확하다고 하지만, 거의 대부분의 소프트웨어를 사용할 때 큰 차이를 느끼질 못한다. 최근엔 400DPI의 볼 마우스도 나오고 있다.

마우스는 원래 미국의 제록스사와 마우스시스템즈에서 시제품을 내놓았지만, PC급에서 지원해 주는 소프트웨어의 부족으로 5~6년전만 해도 사치품으로 인식이 되었었다. 그러던 것이 소프트웨어 업계의 거인격인 마이크로소프트사에서 자사상표의 눈 2개 달린 쥐들을 내놓고, 이후 자사의 소프트웨어들이 그 쥐들을 지원토록 하였다. 그후 마이크로소프트의 쥐들과 호환이 가능한 쥐들이 무섭게 번식하였으며, 국내에서도 이러한 호환 쥐들을 5~14만원대에서 구입할 수 있게 되었다.

본글에서 소개할 각종 함수는 마이크로소프트사의 마우스와 호환이 되는 것을 원칙으로 하였고, 눈의 갯수는 3개로 하였으나, 마지막 버튼을 이용하지 않으면 눈 2개짜리 쥐를 갖고도 완벽하게 수행된다. 최근의 매킨토시 기종이나, PS/2 기종에서는 눈이 1개나 2개 있는 마우스를 기본으로 장착하여 팔만큼 마우스는 제1호 악세사리가 되고 있으니 꼭 구입하시길 권한다.

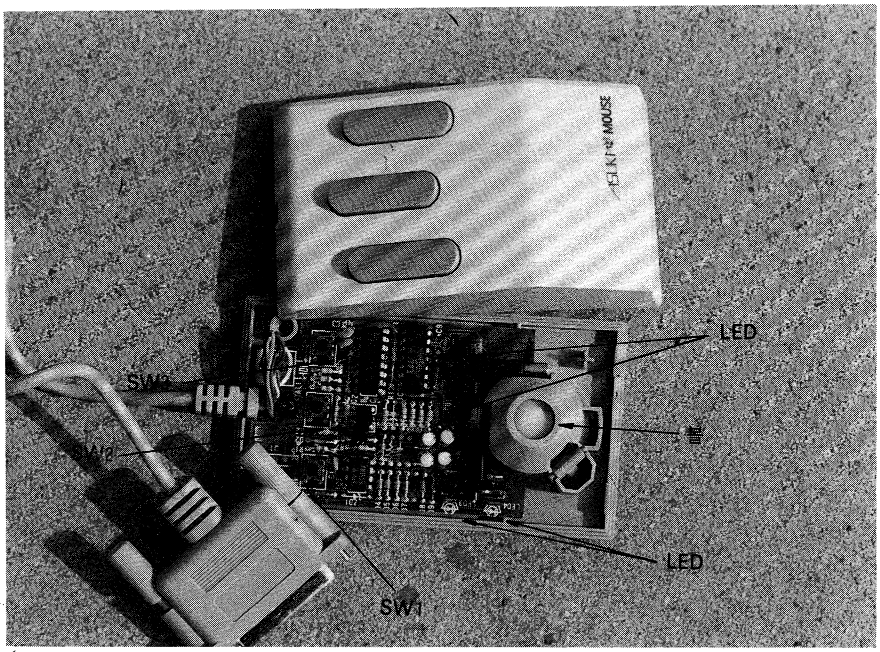
참고로 본글에서 소개하는 내용들과 호환이 가능한 제품은 다음과 같다.

① Microsoft Mouse(2버튼, 볼)–Microsoft Inc.
② PC Mouse(3버튼, 광학)
③ Omni Mouse(2버튼, 볼)–Mouse System Inc.
④ Logi Mouse, Clearcase Mouse(3버튼, 볼)–LogiTech Inc.
⑤ Genius Mouse(3버튼, 볼)–KYE Inc.
⑥ 기타 위 제품과 호환이 가능한 제품들.

결국, 마우스란 직각보행인 키보드와는 달리 사용자가 원하는 위치가 어디든지 즉각적으로 움직일 수 있는 편리한 지시장치이다. 해상도과 낮다는 말이나, 감각도(Sensitivity)가 낮다는 말은 마우스의 움직임에 비해 화면의 커서가 굉장히 급변한다는 의미이다. 따라서 고해상도 마우스는 미세한 부분까지 정확히 이동하기 때문에, 고도의 그래픽 드로잉이나 캐드시스템에서 요구된다.

마우스를 지원하는 소프트웨어

〔사진 1〕 볼마우스의 구조



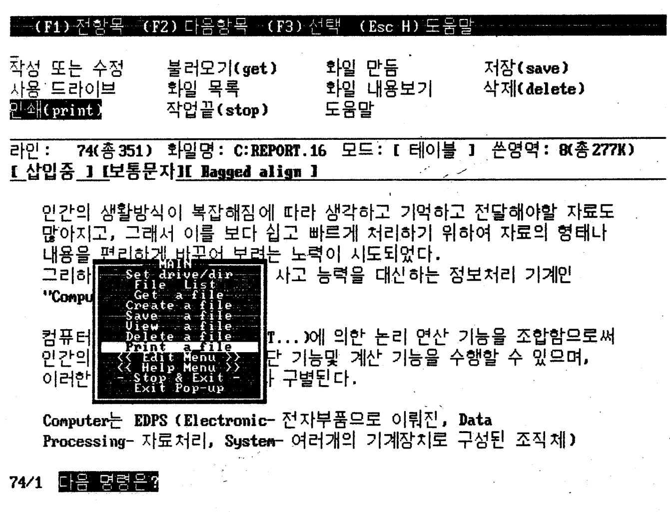
당연한 말이 되겠지만, 컴퓨터는 시키지 않는 일은 절대 못하며, 융통성이 없다. 따라서 마우스를 이용하고 싶어도 소프트웨어 제작회사에서 신경써서 지원하지 않았다면 무용지물이 되고 만다. 그래서 대부분의 마우스 제작회사는 고유의 스크립트 언어를 정하고 사용자가 스스로 자신만의 메뉴디자인 및 키보드 매크로 기능을 이용할 수 있도록 작은 컴파일러를 제공하고 있다. 이러한 메뉴 디자인 제너레이터는 반복되는 키보드 작업을 미리 메뉴로 기억시켰다가 마우스만으로 선택이 가능하게 한 것이다.

화면 1은 보석글에서 자주 사용되는 일들을 메뉴로 짜서 마우스의 선택만으로 키보드를 대신하게 한 팝업(Pop-up) 메뉴(나타났다 사라지는 메뉴)의 예이다. 하지만, 이런 기능은 마우스를 지원하지 않는 소프트웨어들에 있어서는 단순한 메뉴선택 기능만을 지원할 뿐 제약이 많아, 100% 활용할 수가 없다. 이 점을 반드시 기억하여, 프로그램 자체내에서 지원이 가능하게끔 만들어야 하겠다. 현재 매킨토시나 PS/2용의 소프트웨어들은 100% 마우스를 지원하지만, IBM-PC의 경우 약 2만여종의 제품중 7~10%만이 자체 지원된다고 한다. 그리고 저해상도의 CGA 등을 썼던 탓에 개발에 한계가 있었으나, 근래에 고해상도의 그래픽카드의 보급으로 이제는 CAD(Computer Aided Design)나 그래픽 아트 설계 등 그래픽 지향적 제품은 거의 지원되는 상황이다.

이글을 읽으시는 분중에서 소프트웨어 개발에 뜻이 있으신 분은 최근의 추세인 ICON 메뉴(특징있는 작은 그림메뉴) 방식과 WYSIWYG(What You See Is What You Get :입력과 출력이 동일한 방식)방식의 지향적인 프로그램을 마우스 인터페이스함수와 더불어 신경써 주시길 부탁한다.

대표적인 마우스 지원 소프트웨어들은 다음과 같다.

〔화면 1〕 보석글 화면



① **프로그래밍 언어 분야**: Microsoft사의 Quick-C, Quick-Basic, CodeView debugger Windows S.D.K.
Digitalk사의 SmallTalk/V
GSS사의 Graphic Kernel System Lib.
Media Cybernetics사의 Halo Lib, Halo88 Lib.

② **사무 관련분야**: Microsoft사의 Excel, Word, Plan, Works, chart Windows, Xerox사의 Ventura Desktop Publisher
기타, 챠트 및 출판 업무류(PageMaker, Gem, HPG~)

③ **그래픽 관련분야**: Dr. Halo series, PC Paintbrush series 등의 그림작성류 AutoCAD, Versa CAD, OrCAD 등의 캐드 S/W류, Scanner S/W

④ **유틸리티 류**: Fontasy, FontEdit, Norton Commander~
이 프로그램들은 초기화시 마우스를 잘 선택하여야 한다.

마우스를 이용하는 함수들

마우스의 정보 입출력과 제어는 INT 33H를 통한 함수호출(Function Calls)로써 이루어진다. 간혹 INT 33H를 지원해 주는 디바이스 드라이버가 호환이 안되는 경우도 있는데, 이럴 때는 주로 많이 통용되고 있는 시스템화일(ms-mouse. sys)을 이용하면 된다. 이 마우스 디바이스구동 시스템화일은 약 20여종의 함수를 지원하고 있는데, 그중 필자는 사용하기 쉽고 자주 쓰이는 함수 15여종을 추려서 소개하려 한다. 각각의 함수는 다양한 함수로 구현되어 있으므로, 여기서는 중간 의사코드로 함수들을 설명하겠다.

함수 설명을 하기 전에 먼저 알아두어야 할 기초적인 것이 있다. 즉, 모든 마우스의 정보 및 제어는 INT 33H를 거치는데, 다음과 같은 레지스터가 내부에서 이용되고 있다.

AX: 마우스 함수종류 선택 및 마우스 상태 반환
BX: 버튼의 상태에 대한 정보
CX: X축 방향에 관한 관련 정보
DX: Y축 방향에 관한 관련 정보
SD, DI: 마우스의 확장 정보에 쓰임

특히, 마우스의 버튼을 누르고 뗄때 사용되는 용어가 다음과 같이 표준화되어 있으니, 원서를 읽을때 참조하기 바란다.

Press: 버튼을 누른채로 가만히 있는 상태이다.
Release: 버튼에서 손을 뗀 상태이다.
Click: 마우스를 움직이지 않고 잠시 눌렀다 뗀 상태이다.
Chord: 같은 순간에 여러번 눌렀다 뗐다하는 상태를 말함(double click~)

그리고, 마우스의 버튼을 누르면 마우스버튼 상태를 나타내는 2바이트 변

수가 값을 달리한다. 즉 마우스에 대한 개요를 설명했던 부분에서 처럼 각각의 관련 비트가 세트/리세트된다.

가령 왼쪽과 맨 오른쪽 버튼을 동시에 눌렀다면, 리턴되는 BX가 00000000 00000011이 되어 3이 반환된다. 그러나, 실제적으로 동시에 누른다는 것은 힘들다. 왜냐하면, 아무리 동시에 눌렀다해도 어느 것 하나가 먼저 감지되어 값을 반환하기 때문이다. 버튼이 3개인 마우스는 가운데 버튼의 관련 비트를 3번째 비트로 할당하므로써, 호환성을 높혔기 때문에 누르면 값이 4로 된다(00000100). 물론 어느 경우나 뗀 순간부터는 0으로 바뀌겠지만, 상태 체크 함수를 계속 호출한 상태로 놓아두면, 누르는 순간마다 감지되어 작동된다. 이것이 둘다 같이 동작할 수 있는 키 포인트가 된다.

[함수 1] void MsMouse(var m1,m2,m3,m4 :int);

이 함수는 인터럽트 33H를 호출하여 인수들을 전달하고, 다시 되돌려 받는 매개적인 기본함수이다. C의 경우 2가지 스타일로 구현하여 보았으며, 각 메모리 모델에 맞추어서 컴파일하여야 인수전달이 순조로울 것이다. m1..m4는 각각 AX, BX, CX, DX에 전달되어 결과를 받는다.

[함수 2,3] int Herc_ResetMouse(PAGE); 및 int Resetmouse;

두 함수는 모두 마우스를 사용하기 직전에 초기화를 시키기 위해 꼭 호출해야 하는 함수이다. 이 함수는 먼저 msmouse. sys나 msmouse.com이 메모리에 상주되었나를 체크해서 에러가 있으면 0을, 그렇지 않으면 −1을 반환한다.

이 함수를 통해서 마우스는 스크린상의 좌표의 최대, 최소값과 커서의 모양 및 감각도 등을 초기화시켜 동작을 준비한다.

특별히 IBM MDA, CGA, EGA와는 달리 BIOS 수준에서 호환이 안되는 허큘리스 그래픽 카드의 경우 화면의 좌표를 산출하기 위해, 절대번지 0040 : 0049에 담겨있는 CRT 정보를 페이지에 맞추어 변환시켜 주어야 하므로 따로 처리해야 한다.

그리고 B000H 즉 페이지 0의 그래픽에서 마우스를 쓰고 싶으면, Herc_resetmouse(0)을 이용하면 된다. 다시 말하지만, 이 함수는 처음에 꼭 호출되어야만 한다. 만약 텍스트 모드에서 이 함수를 썼다면, 커서는 텍스트 모드의 커서가 되고 움직임도 키보드처럼 캐릭터 사이즈로 움직인다. 그래픽모드에서는 커서가 화살표 모양이 되며, 상하좌우 1비트 단위로 동작을 시킬 수 있다.

참고로, 아까 소개한 보석글상의 화면(640×400)에서 올바른 메뉴 호출을 위해서는 이러한 좌표산출을 위한 교정이 필요하다. 따라서 따로 V_NKP.BIN을 만들었으니, mouse-sys. com을 쓰시는 분은 여기서 제공되는 툴박스를 이용하기 바란다.

EX1) C에서 :

```
    if herc_resetmouse(0) == 0 {
       printf(" Mouse driver is not found. "); beep();
       exit(0);
    }
```

PASCAL에서 :

```
 IF Herc_ResetMouse(0) <> -1 THEN
    BEGIN
      WRITELN(' MOUSE DRIVER IS NOT FOUND! '); HALT;
    END;
```

MASM에서 :

```
              EXTRN  RESETMOUSE:far
mouse_status  dw   ?
              ......
              push AX                ; AX,BX,CX,DX를 저장
              push BX
              push CX
              push DX
              lea  AX,mouse_status   ; mouse_status 의 어드레스를
              push AX                                        저장
              call RESETMOUSE
              pop  DX                ; AX,BX,CX,DX를 재저장
              pop  CX
              pop  BX
              pop  AX
              and  ......            ; mouse_status의 값 체크
```

[함수 4,5] void ShowCursor; void HideCursor;

이 함수는 마우스의 커서를 보이게 하거나, 사라지게 하는 역할을 한다. 마우스의 커서는 3종류가 있다. 하나는 '하드웨어 커서'라고 해서 램상주 드라이버에서 미리 정의된 모양이 셋업된 화면 모드와 카드에 맞춰져 나타나는 것이다.

텍스트모드에서는 꽉찬 직사각형이다. resetmouse를 호출한 후, showcursor를 하거나, settextcursorh 함수를 쓰면 된다. 또, '소프트웨어커서'라고 해서 텍스트모드에서 사용자가 원하는 모양의 커서를 지정하는 것도 있다(참조 [함수 11, 12]).

끝으로 그래픽모드에서 다양한 모습을 나타내기 위한 '비트 맵 커서'가 있는데 화면 2처럼 커서모양을 순서적으로 정의하여 호출하는 것이다. 하지만, 비트 맵 방식은 다음에 소개할 개념을 잘 이해하여야 하며, CGA, EGA 그래픽 모드에서는 1픽셀(점)당 2~4 비트를 점유하고 있어서, 화면 2처럼 단순하지 않다. 즉 CRTC 레지스터를 조정하는 문제도 있고, 큰 사이즈의 커서를 셋업하는데 매우 복잡해 이번 기회에서는 이 부분에 대해선 생략하고 대신 다른 방법으로 다양한 커서를 제작하여 쓰게끔 했다 (참조 demo.pas). 화면 2는 미리 정의되어 디폴트로 사용한 형태이다.

여기서 우리가 잠깐 이해하고 넘어가야할 그래픽 기초이

론이 있다. **화면 3**을 보도록 하자. 도너츠 모양의 이미지와 막대모양이 있다. 이 두모양을 "AND"시키면 공통적으로 1(ON)인 점들만 유지된다.(AND : 1+1만 1인 연산) "OR"을 시키면 합집합처럼 되어 양쪽의 이미지가 중첩된다. 이러한 기법은 흔히 조합형 한글의 표현시 자모형상의 OR화가 한 예이다.(OR : 0+0만 0인 연산)

그리고 우리가 주로 볼 수 있는 것으로 어떤 커서나 이미지가 화면을 왔다갔다 할 때 과연 그 밑의 원상을 그대로 유지하는 식의 연산은 어떤 것일까 하는 의문이 들 것이다. 왜냐하면 AND, OR로는 도저히 다시 원래의 이미지들로 분리하질 못하기 때문이다. 그러한 연산이 바로 "XOR"이다. 애니메이션의 기본이 되는 이 연산의 결과는 **표 2**와 같다.

표 2에서 볼 수 있듯이 1(불이 켜진 상태)의 겹침의 횟수가 홀수일 때 1이 된다. 자, 그럼 **화면 3**의 2번째 경우를 생각해 보자. 겹쳐지는 부분은 0이 된다. 하지만, 이 상태에서 다시 원래의 막대기를 그자리에 XOR를 시켜보라. 그러면, 감쪽같이 도너츠만 남을 것이다.

이러한 기법은 중요하다. 가령 비행접시가 화면위를 돌아다니게 하고 싶으면, 그려진 위치에서 다시한번 XOR을 시켜 비행접시를 지운후 옆으로 움직이고, 다시 XOR를 시키면 이동된 것이 되기 때문이다.

〔화면 2〕 커서모양

```
< Screen Mask >       < Cursor Mask >

1001111111111111      0000000000000000
1000111111111111      0010000000000000
1000011111111111      0011000000000000
1000001111111111      0011100000000000
1000000111111111      0011110000000000
1000000011111111      0011111000000000
1000000001111111      0011111100000000
1000000000111111      0011111110000000
1000000000011111      0011111111000000
1000000000001111      0011111111100000
1000000111111111      0011111000000000
1000100001111111      0010001100000000
1001100001111111      0000001100000000
1111110000111111      0000000110000000
1111110000111111      0000000110000000
1111111011111111      0000000000000000
```

Logical operation

Screen Mask	Cursor Mask	Result bit
0	0	0
0	1	1
1	0	Same as Org.
1	1	Inverse Org.

〔표 2〕

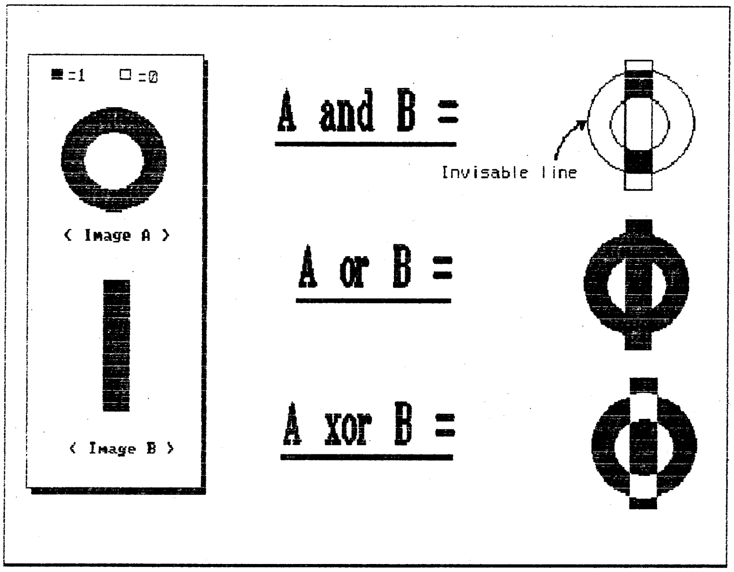
A XOR B		결과
0	0	0
0	1	1
1	0	1
1	1	0

자 그러면, 마우스의 커서가 어떻게 해서 화면에 그려진 원상에 피해를 안끼치고 이동할 수 있는 가를 육안으로 분명히 알 것이다. 그런데, 주의깊게 보면 마우스의 화살표 모양은 언제나 하얗게(ON)보인다. XOR를 한다면 곳에 따라 반전될 때도 있을텐데, 그렇지가 않다. 그 이유는 비트 맵된 커서의 모양이 화면 2에서 볼 수 있듯이 screen mask와 cursor mask 등의 2종류의 모양을 가지고 교묘히(?) 조합한 후 출력되기 때문이다.

먼저, 원상을 기록하고 그위에 screen mask를 원상과 AND 시켜 놓는다. 그 뒤에 cursor mask를 XOR시키면 화살표 모양은 언제나 1이 되어 육안으로 분명히 알 수 있으며, 그 주변의 이미지는 그대로 유지되어 화살표가 정의된 mask block이 무조건 중첩되는 것을 막는다. 그렇기 때문에, 새로운 커서의 모양행렬을 디자인 할 때는 조심해야 한다.

```
[ 함수 5,6 ]  void SetMouseXY(x,y:int);
              int GetMouseXY(var x,y:int);
```

〔화면 3〕 그래픽 연산



이 함수는 마우스 커서의 위치에 관한 정보와 버튼의 상태를 보고해준다. SetMouseXY 함수는 호출값만큼 전달된 X, Y의 좌표로 마우스의 위치를 정해주며, 마우스의 초기화가 이루어진후 부를 수 있다.

GetMouseXY 함수는 매우 중요한 함수로서 현재의 X, Y 좌표값을 call by reference로 전달된 인수에 돌려주고 자기 자신은 버튼의 값을 정의한 2바이트 변수의 값을 돌려준다. (0 : 아무것도 누르지 않은 상태, 1 : 버튼을 누른 상태) 이 함수는 키보드의 어떤 키 입력과는 독립적으로 작동한다고 앞에서 말했듯이, 프로그래밍에 있어서 이점을 주의하여 다음과 같은 형식으로 짜야 계속된 정보를 얻을 수 있다.

가령, 메인 루틴에서 마우스의 좌표를 계속 추적함과 동시에 ESC를 누르면 그 루프를 빠져나오게 하는 부분을 생각해 보자.

EX2) C에서 :

```
for(;;) {
  do {
    BTN=getmousexy(&x,&y);
    if BTN != 0 { check_mouse_btn(); ...  }
     ....
    if (kbdhit()) { ch=get_key();  ... }
     ....
  } while (ch != ESC_CODE);
}
```

PASCAL에서 :

```
REPEAT
  BTN:=GetMouseXY(X,Y);
  IF KEYPRESSED THEN CH:=READKEY;
  IF BTN=1 THEN WRITELN(' LEFT BUTTON PRESSED! ');
  WRITELN(' X: ',x,' Y: ',y);
 UNTIL (CH=#27);
.....
```

```
[ 함수 7,8 ]  void  GetPressInfo( button:int; var  Bstatus,
                          NumbuttonP,OldX,OldY:int ) ;
              void  GetReleaseInfo (  same above ft. );
```

이 함수는 특정 버튼에 대해서 버튼의 상태와 그 버튼을 누르거나 놓았던 횟수 및 가장 최근에 있던 상태의 좌표로 돌려준다. 그리고 가령 윈도우의 메뉴 선택방식인 더블 클릭(Double Click) 등을 감지하는데, 유용하게 쓰일 것이다. 이 함수는 한번 호출될 때마다 카운트가 0이 된다.

```
[ 함수 9,10 ]  void SetMinMaxX( MinX,MaxX :int );
               void SetMinMaxY( MinY,MaxY :int );
```

이 두함수는 마우스가 동작하는 영역을 제어해 주는데 쓰인다. 즉 화면의 일정 캔버스내에서만 동작하고, 그외의 영역에서는 동작하지 못하게 할때 쓰인다. 이때 활동영역은 (MinX, MinY) (MaxX, MaxY)가 이루는 사각영역이다. 그러나, 실질적인 의미에서 이러한 방법보다는 getmousexy된 X, Y좌표가 일정 영역안에 있는지를 체크하여 동작 여부를 정하는 것이 더 나을듯 싶다. (참조 : check_in 함수)

```
[ 함수 11,12 ]  void SetTextCursorH( linestart,linestop :int );
                void SetTextCursorS( ScreenMask,CursorMask :
                                     word );
```

이 함수는 텍스트모드에서 마우스의 커서 모양을 정하는 함수이다. linestart, linestop의 인수는 하드웨어 커서 출력시, 현재의 그래픽 카드가 글자를 생성하는데 주사한 스캔라인의 갯수를 묻는 것이다.

일반적으로 CGA는 0,7이 각각 대응되고, MDA, EGA, HERC에서는 0,11이 대응된다. 함수 11을 resetmouse 후에 부르고 실행시킨후 showcursor를 시키면, 8×12의 블록커서가 나오고, 이동은 각 글자간의 간격에 맞춰져 상하좌우로 이동된다.

한편 함수12의 소프트웨어 커서는 각각의 아스키코드를 대응시켜 커서를 조합하는 것인데, 이 조합을 잘 이용하면 배경색, 블링킹(Blinking), 반전(Reverse) 등의 효과를 낼 수도 있다. 표 3에 소프트웨어 커서를 나타낼 때 참조할 정보를 보였다.

EX3) 가령 강한 밝기의 위방향 화살표 커서를 텍스트커서로 쓰고 싶을 경우

C에서 : …

```
resetmouse();
screenmask=0x0000;   /*  0000h : 현재 값은 */을 무시
cursormask=0x0f18;   /*  8-15 bit: 속성,
                         18h     : 위방향 화살표 코드 */
setextcursors(screenmask,cursormask);
showcursor;
 .......
```

```
[ 함수 13,14 ]  void MotionCounter( var X,Y :int );
                void Sensitivity( Xinc,Yinc :int );
```

이 함수는 마우스의 움직임을 추적하거나, 조절하는 함수이다. 함수 13은 가장 최근에 호출된 직후에서부터 마우스의 위치 변화를 상대적인 좌표값으로 준다. 즉, −32768767안의

〔표 3〕

비트	정 보
15	깜박거림(1), 평상시(0)
14~12	뒷 배경 색깔
11	강도 : 센 밝기(1), 보통(0)
10~8	앞 배경 색깔
7~0	캐릭터의 코드(아스키)

값으로 주는데,(+)값을 준다는 것은 X 값에선 오른쪽으로 간 정도를 의미하고 Y값에선 (+)는 아래로 내려간 상대적인 값을 나타내는 것이다.

한편, 감각도를 의미하는 함수14는 전달하는 Xinc, Yinc의 값이 작을수록 커서가 크게 움직인다. 즉 movement / pixel ratio이기 때문에 화면상의 한 점을 이동시키려면 마우스를 어느 만큼 움직여야 하는 가이다. 가령 (1, 1)이라면, 마우스가 1정도의 움직임이 있어야만 화면의 점 이동이 있으므로, 굉장히 빠른 이동을 할 수 있게 된다.(Xinc, Yinc : 1~100, defalut : (10, 10))

[함수 15] boolean Check_in(x,y,BoxX1,BoxY1,BoxX2,BoxY2 :int);

이 함수는 마우스와 위치가 어떤 영역에 들어 있는가에 대한 여부를 검사한다. (BoxX1, BoxY1, BoxX2, BoxY2)의 영역안에 (X , Y)가 있으면 1, 아니면 0을 돌려준다.

참고로 말하면, 이 함수는 그래픽모드에서만 쓸 수 있다. 텍스트모드에서는 (X, Y) 좌표가 흔히 생각하는 (1, 1,80, 25) 안에 들어있는 값으로 텍스트모드시에 GetMouseXY가 값을 줄것 같은데 그렇지 않다. 그렇다고 1:1 모드인 그래픽모드 좌표로 텍스트 모드에선 돌려주지 않는다. 텍스트모드에선 좌측상단의 글자위치가 (8, 0), 그 옆의 글자가 (16, 0)이고, 둘째줄의 첫째자는 (0, 8) 순으로 되어 있으므로 돌려받는 값의 X값을 8로 나누고 Y도 8로 적당히 나누는 등의 변환이 필요하다.

지금까지의 설명중에서 몇가지 제외된 함수는 있으나, 그래픽모드에서 비트 맵으로 커서 모양을 제어하는 함수는 복잡하며, 컬러에선 몹시 따질 것이 많아 생략하였다. 대신 데모 프로그램에서 손쉽게 커서를 구현시켰으니 참조하기 바란다. 그밖에 라이트펜 에뮬레이트 함수나 마우스 상태 저장, 복구 등의 함수가 있으나 거의 쓸 일이 없고, 호환성이 떨어져 일부러 생략하였다. 필요하신 분은 우편번호 120-749 서울서대문구 연세대학교 공대 전자과 4년 송영길 앞으로 연락하시길 바란다.

그러면 다음에 간략히 언급한 주의사항과 함께 각종 언어로 구현한 소스를 보였다. 도움이 되었으면 좋겠다.

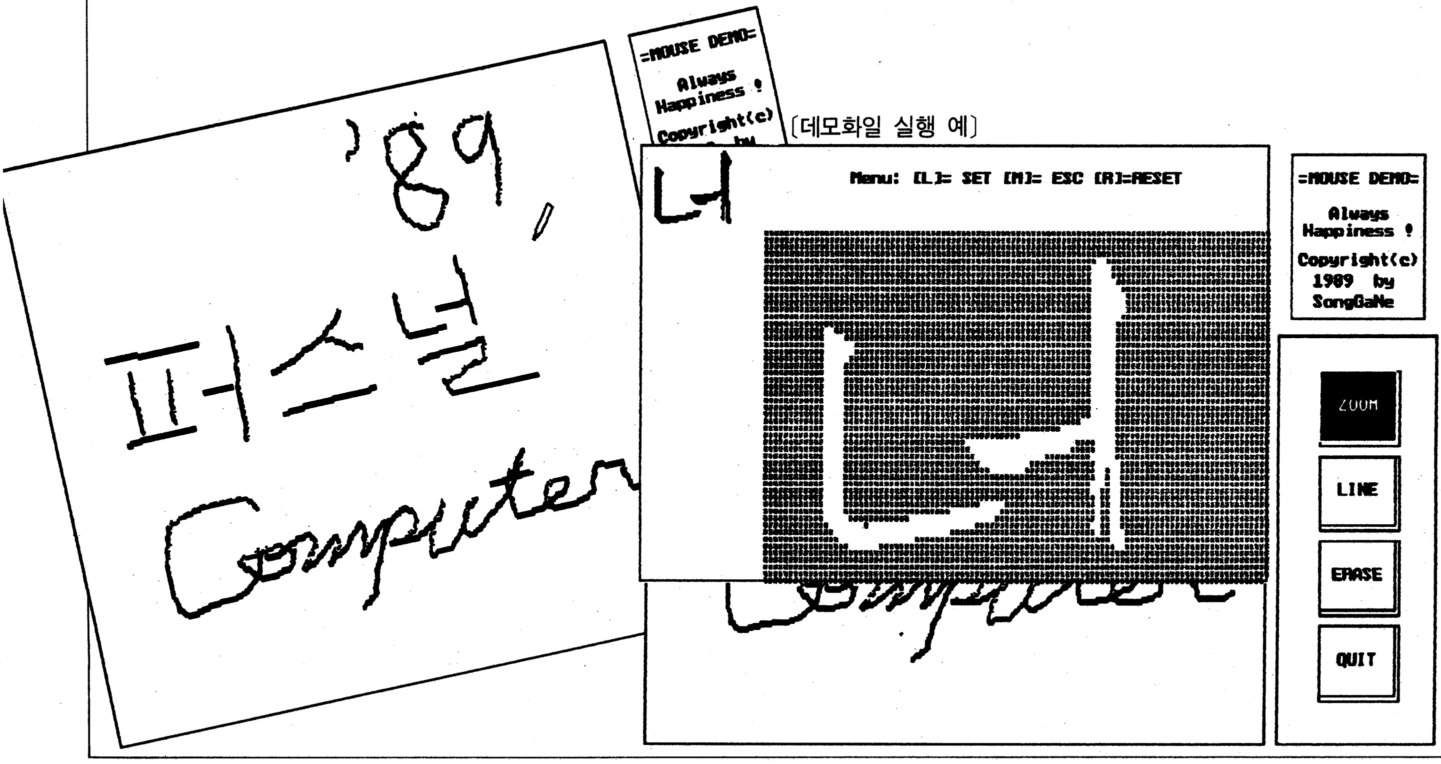
각종 언어로써 구현한 마우스 함수들

화면 4의 리스트는 제공될 화일들의 목록이다. 편의상 각 함수마다 컴파일된 .OBJ .TPU .LIB의 형식도 제공하였다. 다시한번 언급하지만 제공된 함수들을 이용하려면 먼저 메인 메모리에 마우스 드라이버 함수가 상주하고 있어야 한다. 대표적으로 msmouse.sys를 config.sys에 device=msmouse.sys로 등록시켜 부팅하거나, DOS상태에서 msmouse.com을 띄우면 되겠다. 물론 각 제조회사에서 제공하는 것을 써도 호환이 될것이다. 그러나, 간혹 각 회사마다 드라이버 화일이 다르기 때문에 동시에 두종류를 램에 같이 올리면 커서가 깨지는 경우가 있으니 한 종류의 드라이버 화일을 쓰기 바란다.

●Turbo-Pascal 버전 툴키트

전반적으로 평이한 소스이므로 특별히 눈여겨 볼 것은 없

[데모화일 실행 예]



〔화면 4〕 제공되는 함수들과 관련 화일 리스트

```
C:\SONGANE>dir

 Volume in drive C has no label
 Directory of  C:\SONGANE

.            <DIR>      1-01-80  12:08a
..           <DIR>      1-01-80  12:08a
MOUSE_AS ASM      6766  3-19-89   4:25p
EGAVGA   BGI      5363  8-29-88   5:00a
HERC     BGI      6125  8-29-88   5:00a
V_NKP    BIN      2176 12-17-88   1:16a
MOUSE_MS C        3674  3-19-89   2:48p
MOUSE_TC C        4055  3-19-89   2:48p
MSMOUSE  COM      9651  2-06-87  12:00p
MOUSE_AS DOC       715  3-19-89   4:28p
MOUSE_MS DOC       854  3-19-89   2:44p
MOUSE_TC DOC       481  3-19-89   2:46p
DEMO     EXE     25392  4-01-89  10:48p
MOUSE    H        2093  3-19-89   2:10p
MOUSE_AS LIB      1551  3-19-89   4:28p
MOUSE_MS LIB      3087  3-19-89   2:44p
MOUSE_TC LIB      2560  3-19-89   2:46p
MOUSE_AS OBJ       609  3-19-89   4:26p
MOUSE_MS OBJ      2015  3-19-89   2:43p
MOUSE_TC OBJ      1693  3-19-89   2:46p
DEMO     PAS     14592  4-02-89   4:12a
MOUSE_TP PAS      5105  3-19-89   2:56p
MSMOUSE  SYS      7003 10-10-85   2:00a
MOUSE_TP TPU      5328  3-19-89   2:57p
        24 File(s)  22300672 bytes free

C:\SONGANE>
```

다. 단지 이 화일을 자신의 적합한 이름으로 정정한 후 .TPU 화일로 만들어서 사용하면 된다. 허클리스카드 사용자는 ResetMouse 대신 Herc_ResetMouse를 페이지 번호와 함께 사용하면 된다. 만약 여러 종류의 그래픽 카드를 지원하고자 한다면, detect와 detectgraph라는 함수를 이용하여 카드의 종류를 알아낸 후 ResetMouse 함수를 골라서 쓰면 된다(데모 화일 참조).

자세한 이야기는 뒤에 언급한 데모 화일이 파스칼로 짜졌으므로 참조하기 바란다.

● Turbo-C 버전 툴키트

geninterrupt 함수를 이용하여 msmouse(~)를 구현했다. MS-C용의 그것과 비교하기 바란다. 허클리스 사용자는 앞의 내용과 같은 주의사항만 지키면 된다. C는 Pascal과 달리 대소문자를 반드시 구분한다.

mouse.h를 include시켜서 타입 체크를 하게끔 하고, 아울러 각각의 메모리 모델에 맞추어 mouse_tc.c를 컴파일하여 써야 한다. (near far call 때문) 코맨드 라인상에서 mouse-tc.lib를 써넣어 쉽게 컴파일 링크를 할 수도 있지만, 프로젝트 화일에 자신의 프로그램과 더불어 mouse-tc.lib를 등록시키면 된다. 자세한 컴파일 과정은 소스를 보길 바란다.

● Microsoft_C 버전 툴키트

본인이 EGA 카드를 쓰고, Microsoft C가 허클리스 그래픽 모드용 라이브러리가 없고 해서, Herc_resetmouse(int) 함수는 제외했다. 하지만, 만약 원한다면 MS-C에 poke 함수가 없으므로, memory.h의 memset, memmov를 이용해 구현해야 한다.

참고로, MS-C의 그래픽 라이브러리는 골치아프게도 그래픽 X, Y 좌표축을 절대좌표(0, 0~639, 349)가 아닌 논리좌표(−320, −175~320, 175 : 중앙이 0, 0)을 많이 쓰므로 혼란이 온다. 만약 모든 좌표를 Turbo-C 마냥 물리적인 좌표를 쓰고 싶다면 논리좌표의 원점(0, 0)을 화면 중앙에 놓지 말고, 화면 좌측상단에 진짜 (0, 0)으로 잡으면 된다. 그러면 이후의 좌표는 4 사분면의 좌표인양 그냥 받아들이나 결국엔 같은 점이 된다.

● Macro ASeMbler 버전 툴키트

call by value일 경우에는 mov라는 명령어가 주류를 이루겠지만, call by reference일 경우에는 lea 명령을 통해 주소값을 전달하여 변화된 값을 받아야 한다.

자신의 프로그램어에서 쓰일 함수들은 EXTRN…:FAR로 선언한 후 쓰면 된다.

데모 프로그램의 작성

먼저 양해를 구할 것이 있다. 본인의 사정으로 이 데모프로그램을 빠른 시일내로 짜다보니, 기능이 다양하지 못하며 모든 함수를 이용하지 않았다. 또한 본인은 EGA에서 거의 모든 프로그램을 짜기 때문에 그 관련 루틴이 삽입되었다. 물론, 친구의 도움이 있어서 허클리스 카드에서도 이 프로그램은 잘 작동한다. 모쪼록, 마우스를 이용하여 프로그램을 짜는 기본 아이디어만 배운다고 생각하면 좋겠다.

① 프로그램 기능

연필로 선긋는 기능, 지우개로 지우는 기능, 부분확대 편집기능, 다양한 커서모양 변화, 마우스를 이용한 메뉴 선택기법, XOR, AND연산 활용

② 프로그램 실행

- Turbo Pascal로 컴파일(uses mouse_tp.tpu)
- EGAVGA.BGI 또는 HERC.BGI 필요
- 메뉴 선택 및 메뉴 작동은 왼쪽 버튼
- 커서 모양 복귀 및 메뉴 취소는 오른쪽 버튼

• 끝내기는 ESC키 또는 'Q', 'q', ^C 또는 중간 BTN

③ 각 함수들의 설명

```
procedure  CLICK1;  시작 및 정상 상태를 의미하는 소리
           CLICK2;  잘못된 사용시에 나는 소리
           CLICK3;  확인 및 실행을 의미하는 소리

procedure  GET_ZOOM_CUR;
           GET_PEN_CUR;
           GET_ERASE_CUR;
```

이 함수들은 백그라운드 화면(페이지 1)에 앞으로 사용할 커서의 모양을 그린후 전역 변수로 지정된 포인터 변수에 커서 모양을 가리키게 하므로써 차후에 그 모양을 언제든지 신속하게 부를 수 있게 하는 초기화 함수이다.

```
procedure  ZOOM_CURSOR(X,Y);
           PEN_CURSOR(X,Y);
           ERASE_CURSOR(X,Y);
```

이 함수들은 비트 맵 방식이 아닌 그래픽 커서를 구현한 것으로, 일단 원래의 마우스를 숨긴 후 여전히 getmousexy로 얻어진 좌표를 갖고 각각의 커서 이미지를 대신 XOR시켜서 흉내내게 했다. 커서가 움직이기전에 항상 그 위치를 먼저 지우도록 XOR시켜야 하며 플래그를 두어서 초기시에 그려지는 이미지나 마지막의 이미지를 지우도록 하였다. 좌표가 변화하지 않으면 XOR을 안하고 그냥 놔두므로 계속 이미지를 중첩하는 것을 방지하도록 했다. 또한, 연필 모양의 커서는 펜 끝이 (X, Y)를 지적해야 하므로, 커서 이미지를 Y축으로 사이즈 만큼 이동하여 뿌리게 하여야 한다.

```
procedure  INIT_SYSTEM;
           INIT_LOGO_AREA;
           INIT_MENU_AREA;
           INIT_WORK_AREA;
```

이 함수들은 시스템에 맞게 초기화를 시키고, 각각의 영역에 글씨와 메뉴를 썼다. 귀찮더라도 디버깅이나 사이즈, 위치조정을 위해서 많은 const를 두어 좌표를 기억시키는 것이 효율적이다.

```
procedure  INVERSE_BOX(X1,Y1,X2,Y2);
```

이 함수는 전달받은 영역을 메모리에 옮긴 후 XOR을 시켜, 실질적으로 그 사각형을 반전되게 한다. 메뉴 선택의 확인을 위해 마우스가 버튼으로 선택시에 이 함수를 2번 호출하여 깜박거리게 하였다.

```
procedure  FILL_BOX(X,Y);
           EMPTY_BOX(X,Y);
```

이 함수는 산출된 좌표를 갖고 좌우 2점씩 하여 4×4의 블록을 색칠하거나 비우게 하여, 원래의 이미지에서 일어나는 변화를 확대 영역에서도 일치하게 동작시킨다.

```
procedure  BITMAP(X1,Y1,X2,Y2);
```

이 함수는 먼저 확대 편집을 위해 화면의 일부 영역을 저장하여 보호한 후, 선택된 대각선의 좌표가 나타내는 영역을 짤라다가, 맨 좌측 상단에 갖다 놓는다. 그리고 난 후, 그 원상을 훑어 4×4로 표시되는 줌(Zoom)영역에 확대 복사시킨다. 이 함수는 CALC_ZOOMXY, CALC_ORGXY라는 서브함수를 갖고 있어, 원상과 확대 이미지에서의 좌표를 계산하도록 하였다.

각각의 좌표 계산이 끝난 후엔 FILL_BOX, EMPTY_BOX 함수를 불러 확대점에 표시되도록 하였다.

```
procedure  RUN_ZOOM;
           RUN_LINE;
           RUN_ERASE;
           RUN_QUIT;
```

각각의 메뉴 선택시 수행되는 부분이다. 소스를 보고, 탈출 버튼과 키입력을 보길 바란다.

〔리스트 1〕 데모 프로그램

```
{-----------------------------------------------------------}
{                                                           }
{  MOUSE DRIVER TOOL KIT    D E M O  PROGRAM                }
{                                                           }
{                                   By  Song YoungGil       }
{                                                           }
{  TITLE:  THE BABY DR. HALO  ver 1.0                       }
{                                                           }
{-----------------------------------------------------------}
```

```
program MOUSE_DEMO;

uses
   GRAPH,CRT,DRIVERS,MOUSE_TP;
const
   LEFT_BTN    = 1; {마우스 버튼 체크함수에 의해 돌아오는 값}
   RIGHT_BTN   = 2;
   MIDDLE_BTN  = 4;
```

```
   WORK_BOX_X1 = 5;
   WORK_BOX_Y1 = 5;
   WORK_BOX_X2 = 500;
   WORK_BOX_Y2 = 345;
   LOGO_BOX_X1 = 520;
   LOGO_BOX_Y1 = 5;
   LOGO_BOX_X2 = 625;
   LOGO_BOX_Y2 = 100;
   MENU_BOX_X1 = 510;
   MENU_BOX_Y1 = 110;
   MENU_BOX_X2 = 635;
   MENU_BOX_Y2 = 345;
   MENU_CENTER_X = 573;

   ZOOM_MENU_X1  = 543;
   ZOOM_MENU_Y1  = 130;
   ZOOM_MENU_X2  = 603;
   ZOOM_MENU_Y2  = 170;
   LINE_MENU_X1  = 543;
   LINE_MENU_Y1  = 179;
   LINE_MENU_X2  = 603;
   LINE_MENU_Y2  = 219;
   ERASE_MENU_X1  = 543;
   ERASE_MENU_Y1  = 228;
   ERASE_MENU_X2  = 603;
   ERASE_MENU_Y2  = 268;
   QUIT_MENU_X1  = 543;
   QUIT_MENU_Y1  = 277;
   QUIT_MENU_X2  = 603;
   QUIT_MENU_Y2  = 317;
   ZOOM_X1=100; ZOOM_X2=500;
   ZOOM_Y1=50;  ZOOM_Y2=250;
   GW=4; GH=4; {확대 편집시에 사용될 커진 픽셀사이즈}

var
     BTN,X,Y,OX,OY,XX,YY,
     OLDX,OLDY : integer;
     CH     : char;
     FLAG   : boolean; {커서를 그릴때 처음은 XOR 시키면 안됨}
     ZOOM_CUR_P,
     PEN_CUR_P,ERASE_CUR_P : pointer;

procedure CLICK1; {시작할때 나는 소리}
begin
  SOUND(200);
  DELAY(20);
  SOUND(300);
  DELAY(50);
  NOSOUND;
end;

procedure CLICK2; {에러 발생시 소리}
begin
  SOUND(100);
  DELAY(50);
  NOSOUND;
end;

procedure CLICK3; {확인시키려는 소리}
begin
  SOUND(700);
  DELAY(30);
  NOSOUND;
end;

procedure GET_ZOOM_CUR; {점선모양의 확대 커서를 만듬}
var
  SIZE : word;
begin
  SETLINESTYLE(DOTTEDLN,0,NORMWIDTH);
  MOVETO(0,0);
  LINETO(100,0);
  LINETO(100,50);
  LINETO(0,50);
  LINETO(0,0);
  SIZE:= IMAGESIZE(0,0,100,50);
  GETMEM(ZOOM_CUR_P,SIZE);
  GETIMAGE(0,0,100,50,ZOOM_CUR_P^);
end;

procedure GET_PEN_CUR; {기다란 연필을 만듬-좀더 이쁘게 만들 수 있음}
var
  SIZE : word;
begin
  SETLINESTYLE(SOLIDLN,0,NORMWIDTH);
  MOVETO(200,200);
  LINETO(204,194);
  LINETO(217,181);
  LINETO(220,184);
  LINETO(208,196);
  LINETO(200,200);
  SIZE:= IMAGESIZE(200,180,220,200);
  GETMEM(PEN_CUR_P,SIZE);
  GETIMAGE(200,180,220,200,PEN_CUR_P^);
end;

procedure GET_ERASE_CUR; {지우개 커서를 만듬}
var
  SIZE : word;
begin
  SETLINESTYLE(SOLIDLN,0,NORMWIDTH);
  MOVETO(300,100);
  LINETO(330,100);
  LINETO(330,120);
  LINETO(300,120);
  LINETO(300,100);
```

```
  SIZE:= IMAGESIZE(300,100,330,120);
  GETMEM(ERASE_CUR_P,SIZE);
  GETIMAGE(300,100,330,120,ERASE_CUR_P^);
end;

procedure ZOOM_CURSOR(X,Y:integer); {커서를 확대모양으로 바꿔 동작}
begin
  if (X<>OLDX) or (Y<>OLDY) then
    begin
     if FLAG=false then
        PUTIMAGE(OLDX,OLDY,ZOOM_CUR_P^,XORPUT);{첫번째는 생략되어야 함}
    PUTIMAGE(X,Y,ZOOM_CUR_P^,XORPUT);{그래야,커서 모양이 없어짐}
     OLDX:=X; OLDY:=Y;
     FLAG:=false;
    end;
end;

procedure PEN_CURSOR(X,Y:integer); {연필 커서를 나타나게 함}
begin
  if (X<>OLDX) or(Y<>OLDY) then {구좌표와 차이가 없으면 그대로 둠}
    begin
     if FLAG=false then
        PUTIMAGE(OLDX,OLDY,PEN_CUR_P^,XORPUT);
     PUTIMAGE(X,Y,PEN_CUR_P^,XORPUT);
     OLDX:=X; OLDY:=Y;
     FLAG:=false;
    end;
end;

procedure ERASE_CURSOR(X,Y:integer); {지우개 커서가 동작}
begin
  if (X<>OLDX) or (Y<>OLDY) then
    begin
     if FLAG=false then
        PUTIMAGE(OLDX,OLDY,ERASE_CUR_P^,XORPUT);
     PUTIMAGE(X,Y,ERASE_CUR_P^,XORPUT);
     OLDX:=X; OLDY:=Y;
     FLAG:=false;
    end;
end;

procedure INIT_SYSTEM;{마우스의 초기화를 검사된 카드에 맞춰서 함}
var
  GD,GM :integer;
begin
  GD:=DETECT;
  DETECTGRAPH(GD,GM);
  if (GD = EGA) then
   begin
    GD:= EGA; GM:=EGAHI;
    INITGRAPH(GD,GM,'');
    if RESETMOUSE <> -1 then
       begin
        CLOSEGRAPH;
        WRITELN('Mouse driver is not found!');
        HALT(1);
       end
    end;
  if (GD = HERCMONO) then
   begin
    GD:= HERCMONO; GM:=HERCMONOHI;
    INITGRAPH(GD,GM,'');
    if Herc_resetmouse(0) <> -1  then
      begin
       CLOSEGRAPH;
       WRITELN('Mouse driver is not found!');
       HALT(1);
      end;
   end;
  SETMOUSEXY(100,100); {일단, 화면을 (100,1000) 좌표에 놓음}
  CLEARDEVICE;
  SHOWCURSOR; {커서가 나타남}
end;

procedure INIT_LOGO_AREA; {기본적인 로고 화면이 그려짐}
begin
  SETTEXTSTYLE(DEFAULTFONT,HORIZDIR,1);
  SETTEXTJUSTIFY(CENTERTEXT,CENTERTEXT);
  RECTANGLE(LOGO_BOX_X1,LOGO_BOX_Y1,LOGO_BOX_X2,LOGO_BOX_Y2);
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+15,'=MOUSE DEMO=');
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+35,'Always');
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+45,'Happiness !');
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+62,'Copyright(c)');
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+74,'1989  by ');
  OUTTEXTXY(MENU_CENTER_X,LOGO_BOX_Y1+86,'SongGaNe ');
end;

procedure INIT_MENU_AREA; {메뉴화면을 그려줌}
begin
  RECTANGLE(MENU_BOX_X1,MENU_BOX_Y1,MENU_BOX_X2,MENU_BOX_Y2);
  RECTANGLE(ZOOM_MENU_X1,ZOOM_MENU_Y1,ZOOM_MENU_X2,ZOOM_MENU_Y2);
  RECTANGLE(ZOOM_MENU_X1,ZOOM_MENU_Y1,ZOOM_MENU_X2+3,ZOOM_MENU_Y2+3);
  RECTANGLE(LINE_MENU_X1,LINE_MENU_Y1,LINE_MENU_X2,LINE_MENU_Y2);
  RECTANGLE(LINE_MENU_X1,LINE_MENU_Y1,LINE_MENU_X2+3,LINE_MENU_Y2+3);
  RECTANGLE(ERASE_MENU_X1,ERASE_MENU_Y1,ERASE_MENU_X2,ERASE_MENU_Y2);
  RECTANGLE(ERASE_MENU_X1,ERASE_MENU_Y1,ERASE_MENU_X2+3,ERASE_MENU_Y2+3);
  RECTANGLE(QUIT_MENU_X1,QUIT_MENU_Y1,QUIT_MENU_X2,QUIT_MENU_Y2);
  RECTANGLE(QUIT_MENU_X1,QUIT_MENU_Y1,QUIT_MENU_X2+3,QUIT_MENU_Y2+3);
  OUTTEXTXY(MENU_CENTER_X,ZOOM_MENU_Y1+20,'ZOOM');
  OUTTEXTXY(MENU_CENTER_X,LINE_MENU_Y1+20,'LINE');
  OUTTEXTXY(MENU_CENTER_X,ERASE_MENU_Y1+20,'ERASE');
  OUTTEXTXY(MENU_CENTER_X,QUIT_MENU_Y1+20,'QUIT');
end;

procedure INIT_WORK_AREA; {작업영역 초기화하고, 커서의 모양을 준비}
begin
```

```
  RECTANGLE(WORK_BOX_X1,WORK_BOX_Y1,WORK_BOX_X2,WORK_BOX_Y2);
  SETACTIVEPAGE(1);
  GET_ZOOM_CUR;
  GET_PEN_CUR;
  GET_ERASE_CUR;
  SETACTIVEPAGE(0);
end;

procedure INVERSE_BOX(X1,Y1,X2,Y2:integer); {전달된 영역이 반전}
var
  FILLINFO : FILLSETTINGSTYPE;
  P : pointer;
  SIZE : word;
begin
  HIDECURSOR; {커서를 숨기는 이유는 커서의 주위를 보존키 위함}
  GETFILLSETTINGS(FILLINFO); {세팅된 것을 저장}
  SETFILLSTYLE(SOLIDFILL,WHITE);
  SIZE:= IMAGESIZE(X1,Y1,X2,Y2);
  GETMEM(P,SIZE);
  GETIMAGE(X1,Y1,X2,Y2,P^);
  BAR(X1,Y1,X2,Y2);
  PUTIMAGE(X1,Y1,P^,XORPUT); {이때 NOT으로 해도 같다 }
  SETFILLSTYLE(FILLINFO.PATTERN,FILLINFO.COLOR); {구환경 복구}
  SHOWCURSOR;
end;

procedure FILL_BOX( X,Y: integer); {좌표에 맞춰 확대점에 색칠}
begin
  OX := (X - ZOOM_X1) DIV GW;
  OY := (Y - ZOOM_Y1) DIV GH;
  XX := ZOOM_X1 + OX * GW;
  YY := ZOOM_Y1 + OY * GH;
  SETFILLSTYLE(SOLIDFILL,1);
  HIDECURSOR;
  BAR(XX,YY,XX+GW-2,YY+GH-2);
  SHOWCURSOR;
end;

procedure EMPTY_BOX( X,Y: integer); {확대점을 비움}
begin
  OX := (X - ZOOM_X1) DIV GW;
  OY := (Y - ZOOM_Y1) DIV GH;
  XX := ZOOM_X1 + OX * GW;
  YY := ZOOM_Y1 + OY * GH;
  SETFILLSTYLE(SOLIDFILL,0);
  HIDECURSOR;
  BAR(XX,YY,XX+GW-2,YY+GH-2);
  SHOWCURSOR;
end;

procedure BITMAP(X1,Y1,X2,Y2:integer); {사각영역을 확대 편집}
label OUT;
var
  ORG_X1,ORG_Y1,ORG_X2,ORG_Y2,I,J : integer;
  FILLINFO : FILLSETTINGSTYPE;
  P,B : pointer; SIZE : word;
  II,JJ:integer;

  procedure CALC_ZOOMXY( var X,Y : integer); {확대영역의 좌표 산출}
    begin
      X:=100+4*X;
      Y:=50+4*Y;
    end;
  procedure CALC_ORGXY( var X,Y : integer); {원상 좌표로 변환}
    begin
      X:=(X-100) DIV 4;
      Y:=(Y-50) DIV 4;
    end;

begin
  FLAG:=true;
  ORG_X1:=X1;  ORG_Y1:=Y1;  ORG_X2:=X2;  ORG_Y2:=Y2;
  SIZE:= IMAGESIZE(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2); {확대커서가 지적한 곳}
  GETMEM(P,SIZE);
  GETIMAGE(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2,P^);
  SIZE:=IMAGESIZE(0,0,502,252); {확대작업 위해 대비}
  GETMEM(B,SIZE);
  GETIMAGE(0,0,502,252,B^);
  GETFILLSETTINGS(FILLINFO);
 SETFILLSTYLE(SOLIDFILL,BLACK);
 BAR(0,0,502,252);
 PUTIMAGE(1,1,P^,NORMALPUT);
 OUTTEXTXY(300,20,'Menu: [L]= SET [M]= ESC [R]=RESET');
 for I:= 0 to 100 do
   for J:= 0 to 50 do
     if GETPIXEL(I,J)=BLACK THEN
        begin
          II:=I;JJ:=J;
          CALC_ZOOMXY(II,JJ);
          FILL_BOX(II,JJ);
        end;
 SETLINESTYLE(SOLIDLN,0,NORMWIDTH);
 RECTANGLE(0,0,501,251);
 SHOWCURSOR;
REPEAT
  BTN:=GETMOUSEXY(X,Y);
  if KEYPRESSED then CH:=READKEY;
  if (BTN <> 0) then
    begin
     if CHECK_IN(X,Y,ZOOM_X1,ZOOM_Y1,ZOOM_X2,ZOOM_Y2) then
       begin
        if BTN = LEFT_BTN then
          begin
            EMPTY_BOX(X,Y);
            CALC_ORGXY(X,Y);
            PUTPIXEL(X,Y,WHITE);
```

```
        end;
      if BTN = RIGHT_BTN then
        begin
          FILL_BOX(X,Y);
          CALC_ORGXY(X,Y);
          PUTPIXEL(X,Y,BLACK);
        end;
      if BTN = MIDDLE_BTN then  goto OUT;
      end;
    end;
UNTIL ((CH=^C) or (CH=#27)); {ESC키,^C 또는 Middle버튼을 누름}
  CH:=' '; {빠져 나가더라도, 메인루프에서 계속 빠져 나가는 것을 방지}
OUT:
  SETMOUSEXY(10,70); {손상되지 않은 상태로 잠시 저장}
  HIDECURSOR;
  GETIMAGE(1,1,101,51,P^);
  PUTIMAGE(0,0,B^,NORMALPUT); {대피했던 작업영역을 복구}
  PUTIMAGE(ORG_X1,ORG_Y1,P^,NORMALPUT);
  CLICK3;
  INVERSE_BOX(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2); {확대된 곳을
  INVERSE_BOX(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2);  확인시킴}
  INVERSE_BOX(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2);
  INVERSE_BOX(ORG_X1,ORG_Y1,ORG_X2,ORG_Y2);
  SETFILLSTYLE(FILLINFO.PATTERN,FILLINFO.COLOR);
end;

procedure RUN_ZOOM; {확대작업에 관한 프로시듀어}
LABEL BREAK;
begin
  HIDECURSOR;
  BTN:=0;
  FLAG:=true;
  repeat
    ZOOM_CURSOR(X,Y);
    BTN:= GETMOUSEXY(X,Y);
    if ((BTN=LEFT_BTN) and
       (CHECK_IN(X,Y,WORK_BOX_X1,WORK_BOX_Y1,WORK_BOX_X2,WORK_BOX_Y2)))
     then begin
          PUTIMAGE(X,Y,ZOOM_CUR_P^,XORPUT);
          BITMAP(X,Y,X+100,Y+50);
          GOTO BREAK;
         end;
  until (BTN=RIGHT_BTN);
  PUTIMAGE(X,Y,ZOOM_CUR_P^,XORPUT); {확대 안하고 취소할 때
BREAK:                                 커서지움}
  SHOWCURSOR;
end;

procedure RUN_LINE; {줄긋는 작업을 하는 프로시듀어}
begin
  HIDECURSOR;
  BTN:=0;
  FLAG:=true;
  CH:=' ';
  SETLINESTYLE(SOLIDLN,0,THICKWIDTH);
  repeat
   if KEYPRESSED then CH:=READKEY;
    BTN:= GETMOUSEXY(X,Y);
    PEN_CURSOR(X,Y-21); {연필커서를 위에 올려 그려야 모양이 맞다}
    if BTN=0 then MOVETO(X-1,Y); {커서모양이 방해하지 않도록 바꿈}
    if ((BTN=LEFT_BTN) and
        (CHECK_IN(X,Y,WORK_BOX_X1,WORK_BOX_Y1,WORK_BOX_X2,WORK_BOX_Y2)))
        then  LINETO(X-1,Y);
  until ((BTN=RIGHT_BTN) or (CH=#27));
  PUTIMAGE(X,Y-21,PEN_CUR_P^,XORPUT); {마지막 남은 커서도 지움}
  SHOWCURSOR;
end;

procedure RUN_ERASE; {지우는 작업을 관장}
begin
  HIDECURSOR;
  BTN:=0;
  FLAG:=true;
  repeat
   if KEYPRESSED then CH:=READKEY;
    BTN:= GETMOUSEXY(X,Y);
    ERASE_CURSOR(X,Y);
    if BTN=0 then MOVETO(X,Y);
    if ((BTN=LEFT_BTN) and
        (CHECK_IN(X,Y,WORK_BOX_X1,WORK_BOX_Y1,WORK_BOX_X2,WORK_BOX_Y2)) and
        (CHECK_IN(X+30,Y+20,WORK_BOX_X1,WORK_BOX_Y1,WORK_BOX_X2,WORK_BOX_Y2)))
        then  PUTIMAGE(X,Y,ERASE_CUR_P^,ANDPUT); {빈이미지를 AND시켜 지움}
  until ((BTN=RIGHT_BTN) or (CH=#27));
  PUTIMAGE(X,Y,ERASE_CUR_P^,XORPUT); {마지막 커서를 화면에서 지움}
  SHOWCURSOR;
end;

procedure RUN_QUIT; {끝낸다}
begin
  CH:='Q';
end;

begin   (**************** 메인 루틴 ******************)

  INIT_SYSTEM;
  INIT_LOGO_AREA;
  INIT_MENU_AREA;
  INIT_WORK_AREA;
  BTN:= 0; {만약을 대비 초기화를 꼭 한다}
  CH:=' ';
  CLICK1;

  repeat
    BTN:=GETMOUSEXY(X,Y); {키보드와는 별개이므로 함수 계속 체크}
    if (BTN <> 0 ) then  {버튼 눌렀을때만 감지}
     begin
```

```
      if (BTN=LEFT_BTN) then
       begin
        if (CHECK_IN(x,y,ZOOM_MENU_X1,ZOOM_MENU_Y1,ZOOM_MENU_X2,ZOOM_MENU_Y2))
          then begin
            INVERSE_BOX(ZOOM_MENU_X1,ZOOM_MENU_Y1,ZOOM_MENU_X2,ZOOM_MENU_Y2);
            CLICK3; RUN_ZOOM;
            INVERSE_BOX(ZOOM_MENU_X1,ZOOM_MENU_Y1,ZOOM_MENU_X2,ZOOM_MENU_Y2);
          end;
        if (CHECK_IN(x,y,LINE_MENU_X1,LINE_MENU_Y1,LINE_MENU_X2,LINE_MENU_Y2))
          then begin
            INVERSE_BOX(LINE_MENU_X1,LINE_MENU_Y1,LINE_MENU_X2,LINE_MENU_Y2);
            CLICK3; RUN_LINE;
            INVERSE_BOX(LINE_MENU_X1,LINE_MENU_Y1,LINE_MENU_X2,LINE_MENU_Y2);
          end;
        if (CHECK_IN(x,y,ERASE_MENU_X1,ERASE_MENU_Y1,ERASE_MENU_X2,ERASE_MENU_Y2))
          then begin
            INVERSE_BOX(ERASE_MENU_X1,ERASE_MENU_Y1,ERASE_MENU_X2,ERASE_MENU_Y2);
            CLICK3; RUN_ERASE;
            INVERSE_BOX(ERASE_MENU_X1,ERASE_MENU_Y1,ERASE_MENU_X2,ERASE_MENU_Y2);
          end;
        if (CHECK_IN(x,y,QUIT_MENU_X1,QUIT_MENU_Y1,QUIT_MENU_X2,QUIT_MENU_Y2))
          then begin
            CLICK3; RUN_QUIT;
          end;
       end;
      if (BTN=RIGHT_BTN) then
        begin
         CLICK1; {메인루틴에서는 지원하지 않음}
        end;
      if (BTN=MIDDLE_BTN) then
         CLICK2; {호환성을 위해 가운데 버튼 제외}
    end; (* IF BUTTON PRESSED *)
   if KEYPRESSED then  CH:=READKEY; {키보드에 어떤 입력이 있을때}
  until ((CH='Q') or (CH='q') or (CH=^C) OR (CH=#27));

  CLOSEGRAPH;
end.
```

하드디스크에서의 파킹

이상훈 /시게이트 기술과장

이제 하드디스크는 기본적인 보조기억매체로서 널리 인식되고 있다. 그러나 플로피디스크 드라이브를 쓰다가 처음으로 하드디스크를 접하는 경우 기본적으로 알아두어야 할 몇가지 지식들을 몰라 문제가 발생할 때가 많다. 하드디스크의 디렉토리 관리, 파티션 설정 등 소프트웨어적인 사항 등을 비롯하여 포매팅 작업과 하드디스크의 관리에 필요한 하드웨어적인 지식들이 그것들인데 여기서는 하드디스크의 관리에 필요한 디스크 파킹(Parking)에 대해서 알아 보기로 하자.

하드디스크 드라이브의 헤드는 전원이 가해지는 경우 디스크에서 매우 작은 간격을 유지하며 공중에 떠있는 상태로 디스크를 읽고 있다. 이 간격을 플라잉 헤이트(Flying Height)라고 부르며 흔히들 담배연기의 입자 하나가 통과할 정도의 미세한 간격이라고 하는데 사실이다.

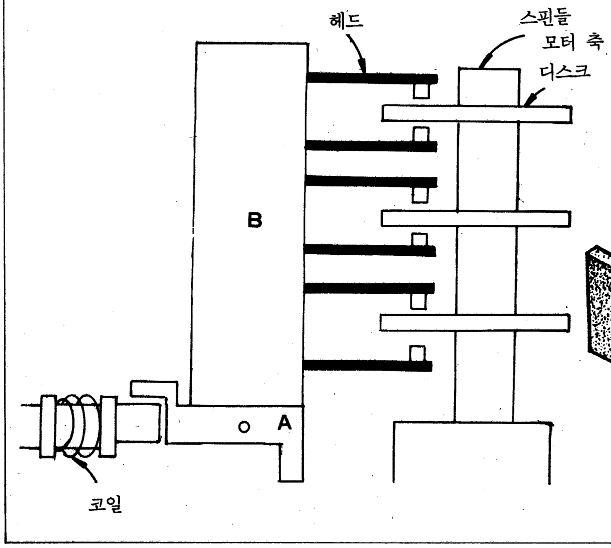
전원이 꺼지는 경우에는 하드디스크 드라이브의 종류에 따라 헤드의 상태가 2가지로 구분된다. 즉 헤드가 동작하던 디스크의 임의의 부분에 그대로 위치한 채 디스크면에 내려앉는 것과 헤드가 디스크의 특정 영역(데이터 영역으로 쓰이지 않는 부분)으로 자동적으로 이동되는 형태가 있다. 이렇게 디스크의 특정영역으로 헤드가 자동적으로 이동되는 것을 오토 파킹(Auto Parking)이라고 부른다. 오토파킹이 되지 않는 하드디스크 드라이브에서는 강제로 헤드를 특정영역으로 이동을 시켜야 하는데 이것을 파킹 또는 쉬핑(S-hiping)이라고 한다.

오토파킹이 되지 않는 드라이브에서 파킹을 시키지 않으면 헤드가 디스크의 데이터 영역과 접촉되어 있는 상태로 있기 때문에 외부에서 가해질 수 있는 충격에 의해서 디스크 또는 헤드에 손상을 입게 되고 이렇게 되면 데이터를 읽을 수 없는 상태로까지 될 수 있다. 물론 하드디스크 드라이브가 들어 있는 PC에 충격을 가하지 않으면 아무런 문제가 되지 않으나 언제 누구에 의해서 어떻게 충격이 가해질 지 모르기 때문에 항상 파킹을 시켜두는 습관을 몸에 지니는 것이 좋다.

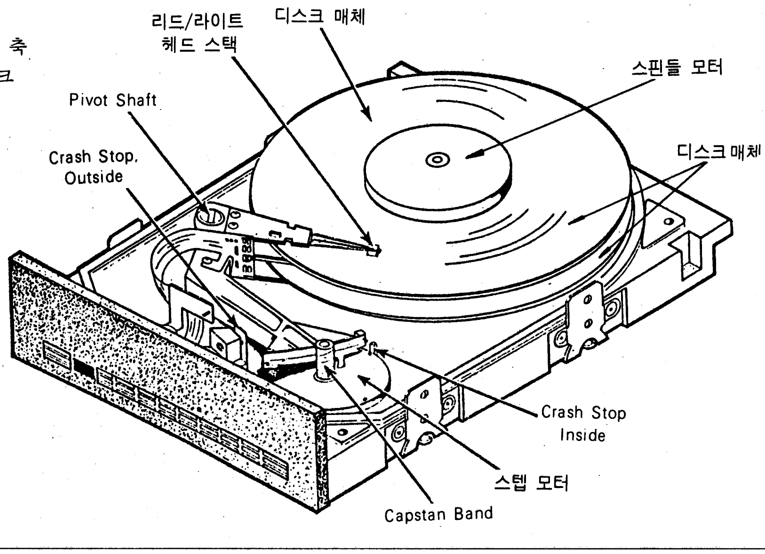
오토 파킹의 종류

여러가지 방법에 의해서 오토 파킹을 시킬 수 있으나 여기서는 시게이트(Seagate)의 제품에 기준하여 설명하고자 한다.

〔그림 1〕 기계적인 오토 파킹 원리



〔그림 2〕 스텝 모터를 이용하는 하드디스크(모델 : Seagate ST-225)



●기계적인 오토 파킹

보이스 코일을 이용하는 하드디스크 드라이브에서 이용된다.

그 구조는 그림 1에 보인 바와 같다.

그림 1은 전원이 꺼져있는 상태이며 하드디스크에 전원이 가해지면 A가 전자력에 의해 뒤로 이동하며 헤드를 구동시키는 B가 자유롭게 되어 스텝 펄스(Step Pulse)가 가해짐에 따라 원하는 동작을 할 수 있게 된다. 이러한 기계적인 자물쇠(Lock Assembly)에 의하는 경우는 헤드가 파킹되고 또 파킹에서 빠져나와야 하므로 전원이 가해지거나 꺼질 때 하드디스크 드라이브에서 둔탁한 소리가 난다.

●전기적인 오토 파킹

스텝퍼 모터(Stepper Motor)를 이용하는 하드디스크 드라이브에서 이용하는 방법이다.

즉 전원이 꺼지는 순간에 특정한 회로의 도움으로 헤드를 이동시키는 것이다(그림 2 참조).

이 경우도 전원을 끌 때 이상한 소리가 발생할 수 있으나 이것은 파킹이 되고 있다는 것이므로 걱정할 필요는 없다. 그림 2를 보면 크래시 스톱 인사이드(Crash Stop Inside)라고 하는 것이 보이는데 이것이 파킹시에 헤드를 구동시켜주는 장치와 부딪쳐서 소리가 난다.

파킹 프로그램

현재 많이 사용되는 하드디스크 드라이브 중에서 ST225, ST238R은 오토파킹이 되지 않는 것이며 이와 같이 오토 파킹 기능이 없는 드라이브를 위해서 많은 회사에서 프로그램을 제공하고 있다. 삼보컴퓨터의 TGSHIP이 이런 프로그램이며 Ontrack사의 디스크 매니저도 Diskpark라는 유틸리티를 제공하고 있다.

한편 콘트롤러 회사에서도 콘트롤러내에 파킹 프로그램을 내장시키는 경우가 있는데 한 예로 시게이트의 ST11R 또는 ST11M을 사용하는 경우 INT 13H(AH=55H, AL=15H)를 이용하면 드라이브를 파크시킬 수 있다. 간단하게 이 프로그램을 DEBUG를 이용해서 만들어 보면 화면 1과 같다.

이 경우는 특정한 콘트롤러를 사용하는 경우이며 그렇지 않은 경우는 화면 2의 프로그램을 이용하면 된다. 프로그램 중에 'RET'가 두번 나타나는데 첫번째는 Drive 2를 수행하기 위한 것이며 두번째는 DOS로 빠져나오기 위한 것이다.

〔화면 1〕 DEBUG를 이용한 간단한 파킹 프로그램

```
N ST11PARK.COM
A 100
MOV AH,55   ;Seagate Extended Interrupt 13H function
MOV AL,15   ;파킹 평션
MOV DL,80   ;첫번째 하드디스크 드라이브
INT 13
RCX
8
W
Q
```

〔화면 2〕 콘트롤러 지원이 없는 경우의 파킹 프로그램

```
N PCPARK2.COM
A 100
MOV  DX,0080       ;첫번째 하드디스크 드라이브
CALL 0109          ;두번째 드라이브 파킹 루틴
MOV  DX,0081       ;두번째 하드디스크 드라이브
MOV  AH,08         ;Get parameter
PUSH DX
INT  13            ;INT 13 수행후 CH,CL은 실린더 수를
                    포함하게 된다.
MOV  DX,0137
JB   011D          ;에러가 발생할 경우 'Park Not supported'
                    메시지를 화면에 표시
MOV  AH,0C         ;seek function
POP  DX            ;drive ID를 다시 DX에 입력
INT  13            ;Get parameter에 의해서 얻은 마지막
                    실린더 값으로 헤드를 이동시킴
JB   0125          ;에러가 발생하면 두번째 drive로 이동
MOV  DX,0126       ;Drive 1 parked라는 메시지를 화면에 표시
MOV  AH,09
INT  21
INC  BYTE PTR [012C] ;Drive 2 parked라는 메시지를 화면에
                    표시하기 위한 준비작업
RET
DB  'Drive 1 parked',0D,0A,'$'
DB  'Park Not supported',0D,0A,'$'
RCX
4C
W
```

파스칼에서의 지수계산

파스칼에서 지수를 어떻게 처리할지 고민하는 유저들이 있다면 이 글이 도움이 되길 바란다. 포트란이나 베이직에서는 매우 복잡한 지수계산도 X**Y, X^Y 등으로 쉽게 할 수 있으나 파스칼에서는 X의 제곱(X^2)인 SQR(X)이나 X의 제곱근($\sqrt{X}$)인 SQRT(X)와 같은 기능을 제외하면 지수계산 기능이 거의 없다고 볼 수 있다.(이러한 기능을 갖춘 MS-PASCAL 과 IBM 파스칼은 예외이다)

따라서 여기에서는 터보파스칼상에서 X가 실수이고 Y가 실수이거나 정수일 경우, X값에 대한 지수를 계산하는 2가지 기능을 소개한다. 하나는 **리스트 1**의 PWRR (X, Y)인데 이는 X가 실수이고 Y가 실수로 실수지수인 경우의 기능이고, 또 다른 하나는 **리스트 2**의 PWRI(X, N)인데 이는 N이 정수이고 X는 실수로 정수지수인 경우의 기능이다. 그러나 이 기능은 반드시 하나가 라이트된다 할지라도 정수제곱에서의 정수에 대해 특별히 마련된 것은 없다.

예를 들어 2^{-2}에 대한 지수계산을 리스트 1과 리스트 2를 통해 실행시키면 그 결과치는 0.25가 된다. 만약 정수계산이 요구되면 이 결과는 0이 된다. 왜냐하면 PWRI는 절대치가 1미만인 모든 결과는 0으로 취급하기 때문이다. 독자들은 여기에서 제공하는 아이디어를 이용하여 이 경우에 대한 적절한 알고리즘을 라이트해 보기 바란다.

만약 Y가 실수이면 PWRR : =EXP(Y*LN(X))를 사용하여야만 한다. 그러나 이 식은 Y가 실수이면 모두 사용할 수 있는 것은 아니다. X<0이면 LN(X)는 복소수(Complex Number)가 되는데 파스칼에서는 복소수형은 계산이 불가능하다. 그리고 X=0인 LN(0)은 정의되어 있지 않으므로 0Y의 경우는 개별적으로 처리해야 된다. 먼저 0^0는 정의되어 있지 않으므로 하나는 제외된다. 만일 Y>0면 모든 Y에 대해 0Y는 0이 된다(0Y=0).

그러나 중요한 경우는 X=0이면 Y<0인 경우이다. 왜냐하면 X^{-Y}의 경우, 이는 $1/(X^Y)$가 되기 때문에 0으로 나누게 되는 결과가 되고 이 결과는 무한대(∞)가 되어 컴퓨터가 거부하는 상황이 된다.

리스트 1의 PWRR(X, Y) 기능을 **리스트 3**의 드라이버 프로그램을 통해 실행시켜 보면 X, Y의 값에 대한 시험결과가 앞에서 설명했던 내용과 일치하고 있다는 것을 알 수 있다.

그리고 이 리스트 1에서 Y는 실수로 정의되어 있지만 이 Y가 정수일 경우에는 계산속도가 더 빨라진다. 그러나 LN 및 EXP 계산은 상대적으로 늦어지는데 이 이유는 절단무한행렬(Truncated Infinite Series)로 계산하기 때문인데 이 절단무한행렬에는 절단오차(Truncation Error)의 보정을 위한 일부 보정계수가 첨가되어 있다. 이 알고리즘에서는 지수의 2진화 표현을 오른쪽에서 왼쪽으로 스캔하여 최적계산시간(Near-Optimum Computation Time : 리스트 2)을 갖는 제곱 및 곱셈수행형태를 만든다.

예를 들어 $X^5=(X*X)^2*X$의 식을 사용할 경우에는 단지 3번의 곱으로 계산이 가능하지만 $X^5=X*X*X*X*X$ 식에 의한 계산은 4번의 곱을 하게 되므로 그만큼 계산시간이 더 걸리게 된다. 따라서 지수계산에 의한 방법을 이용하는 것이 계산속도를 빨리할 수 있다는 것을 알 수 있다. 그러나 이 알고리즘은 10이하의 적은 지수에 대해서는 많은

〔리스트 1〕 PWRR.PAS

```
FUNCTION PWRR(X,Y:REAL): REAL;
BEGIN
  IF X>0 THEN PWRR :=EXP(Y*LN(X))
  ELSE IF X<0 THEN
    BEGIN
      WRITELN('X<0. Halt.');
      HALT
    END
  ELSE IF (X=0) AND (Y=0) THEN
    BEGIN
      WRITELN('0 to the 0 power. Halt.');
      HALT
    END
  ELSE IF (X=0) AND (Y<0) THEN
    BEGIN
      WRITELN('0 to a negative power. Halt.');
      HALT
    END
  ELSE PWRR:=0.0
END;
```

〔리스트 2〕 PWRI.PAS

```
{
+------------------------------------------------+
|                    PWRI.PAS                     |
+------------------------------------------------+
}

FUNCTION RLSCAN(X: REAL; N:INTEGER): REAL;

VAR   Y,Z:REAL;
        O:BOOLEAN;
     BIGN:INTEGER;

BEGIN
  BIGN := N; Y :=1.0;  Z := X;
  WHILE BIGN > 0 DO
    BEGIN
      O := ODD(BIGN);
      BIGN := BIGN DIV 2;
      IF O THEN
        BEGIN
           Y := Y*Z;
           RLSCAN := Y
        END;
      Z := Z*Z
    END;

END;

FUNCTION PWRI(X: REAL; N : INTEGER): REAL;

 BEGIN
  IF (N>0) THEN PWRI := RLSCAN(X,N)
  ELSE  IF (X<>0.0) AND (N<0) THEN
    BEGIN
       N:=-N;
       PWRI :=1.0/RLSCAN(X,N)
    END
  ELSE IF (N=0) AND (X<>0) THEN PWRI := 1.0
  ELSE IF (N=0) AND (X=0) THEN
    BEGIN
      WRITELN('0 to the 0 power, halt,');
      HALT
    END
  ELSE IF (N<0) AND (X=0) THEN
    BEGIN
      WRITELN('Division by zero. halt.');
      HALT
    END
  END;
```

〔리스트 3〕 PWRRTST.PAS

```
PROGRAM PWRTST;
VAR
 X,Y,Z : REAL;

{$I PWRR.PAS}

BEGIN
 WRITELN('Enter X and Y');
 READLN(X,Y);
 Z := PWRR(X,Y);
 WRITELN(Z)
END.
```

〔리스트 4〕 PWRITST.PAS

```
PROGRAM PWRITST;

/AR
 Z,X: REAL;
   N:INTEGER;

{$I PWRI.PAS}

BEGIN
  WRITELN(' Input X,N');
  READLN(X,N);
  WRITELN;
  WRITELN(Z);
END.
```

시간을 절약시키지 못하고 큰 지수의 계산에서는 실행속도를 아주 빨리할 수 있어 시간이 많이 절약된다.

리스트 2에서 X<0이고 N>0은 허용되지만 0은 역시 정의되지 않는다. 앞에서와 같이 X=0이고 N<0은 0으로 나누는 것을 의미하므로 제외되어야 된다. 그리고 X=0이고 N>0이면 결과는 0이 된다. X>0이고 N<0이면 비트탐색 알고리즘을 ABS(N)에 적용하고 다시 원래대로 교환(Reciprocation) 시켜 주어야 된다. PWRI(X, N)은 리스트 2에서 볼 수 있듯이 터보 파스칼에서도 라이트된다.

그러나 이들 두 기능에는 오버플로우에 대한 어떠한 대책도 마련되어 있지 않기 때문에 만약에 1000^{1000}을 계산하려고 하면 루틴에러가 발생된다. 오버플로우되지 않은 경우에 있어서 이 기능들은 아주 간단명료하게 결과치를 보여준다.

리스트 3과 리스트 4는 PWRR(X, Y)와 PWRI(X, N) 기능을 시험하는 드라이버 프로그램이다.

마우스를 이용한 그래픽에디터
SEM-DRAW

박세희/중앙대 셈틀연구회

다음에 소개하는 SEM-DRAW 프로그램은 일종의 마우스를 이용한 그래픽에디터로서 마우스를 유용하게 사용할 수 있게 해준다. 따라서 여기에 있는 소스 프로그램을 제대로 이용한다면 각자가 나름대로 필요한 루틴 등을 첨가해 직접 마우스 프로그램을 작성할 수 있을 것이다.

들어가는 말

컴퓨터를 만지는 사람이라면 누구나 한번쯤 그래픽에 관심을 가져 보았을 것이다. 특히 요즘은 마우스를 그래픽에 이용하는 사례가 인기를 모으고 있는 실정이며, 이는 우리 주변에서 쉽게 찾아 볼 수 있는 오토 캐드(Auto CAD), 닥터 할로우(DrHalo) 등 거의 모든 그래픽 패키지에서 볼 수 있는 현상이다. 이같이 마우스가 인기를 얻는 것은 신속한 메뉴 선택과 화면의 임의의 위치를 선택해서 재빠르게 원하는 도형을 그려줄 수 있는 장점을 갖고 있기 때문일 것이다.

필자는 이런 마우스를 이용해서 프로그램을 작성하고자 하는 독자들을 위해 마우스 환경 및 예제 프로그램을 소개하고자 한다. 여기에 설명되는 내용은 대체로 국내에 보급이 많이 된 허큘리스카드를 위주로 했으며 따라서 뒤에 소개하는 그래픽 에디터도 허큘리스에 맞게 프로그래밍 하였다.(허큘리스를 사용하지 않는 독자들에게는 죄송하지만 필자가 허큘리스 카드를 사용하고 있는 관계로 양해를 구한다.)

그리고 여기에 실린 프로그램 소스는 독자들의 이해를 돕기 위해 동일한 유형의 루틴을 생략하여 프로그램의 길이를 줄였다. 여기의 루틴을 이해한 독자라면 충분히 나름대로의 취향에 맞게 루틴들을 첨가하여 더욱 훌륭한 프로그램으로 발전시킬 수 있으리라 믿는다. (이것은 또한 관심있는 독자들이 직접 마우스를 이용해서 프로그램을 작성하는 데 참고가 되어 주었으면 하는 필자의 바램에서이다.)

한편 마우스에 관한 내용은 앞글에서 송 영길씨가 자세하게 설명하였기 때문에 여기에서는 생략해 버리고 단지 기본적으로 알고 있어야 할 사항들만 설명하기로 하겠다.

필요한 장비

① IBM-PC XT / AT 또는 호환기종
② RS-232C 시리얼 카드
③ 마우스
④ 마우스 드라이버 프로그램
(마우스를 구입할 때 관련 소프트웨어는 같이 얻을 수 있지만 RS-232C 시리얼 카드는 별도로 구입을 해야 한다.)

마우스 사용상의 기본 지식

(1) 마우스의 종류

마우스를 사용하고 있는 유저라면 익히 알고 있는 내용이다. 마우스에는 크게 2가지가 있는데 마우스 바닥에 공(Ball)이 있는 볼 마우스(Ball Mouse)와 바닥에 발광 다이오드가 있는 옵티컬 마우스(Optical Mouse)가 그것이다.

(2) 화면

컴퓨터의 그래픽을 지원하고 있는 카드는 여러 종류가 있지만 여기서는 허큘리스의 경우를 다룬다. 다른 그래픽카드에는 별 문제가 안되지만 허큘리스의 경우 그래픽 상태에서 그대로는 커서가 나타나지 않는다는 특징이 있는데 이는 조금 후에 설명되므로 일단 여기서는 마우스 드라이버가 현재 사용되고 있는 그래픽지원 카드를 체크하여 커서의 색이나 도트수를 계산하여 화면에 표시하므로 사용자에게는 문제가 아님을 알아 두자.

허큘리스 그래픽카드는 2개의 화면을 제공하고 있다. 즉 0번과 1번 페이지(0번인 경우는 B0000H~B7FFFH, 1번인 경우는 B8000H~BFFFFH의 메모리에 있다)를 말하는데 앞서 말한 그래픽 상태에서 커서를 나타나게 하려면 40H:49H번지에 페이지 0인 경우는 6을, 페이지 1인 경우에는 5를 넣어 주면 된다. 그러나 SIMCGA나 HBASIC을 사용한 경우는 커서가 잘 나타난다. 문제는 터보 C를 사용하는 경우인데 이 경우는 다음 순서를 따르면 별 문제가 없을 것이다.

① 허큘리스를 그래픽 모드로 세트한다.
② 40H:49H에 해당 수치를 입력한다.
③ 마우스를 초기화시킨다. 초기화는 마우스 함수 0번을 호출하면 되는데 이것은 조금 후에 설명한다.

(3) 커서

마우스에는 다음과 같이 3종류의 커서가 준비되어 있다.

① 그래픽 커서(화살표나 손가락 또는 십자 모양)

② 소프트웨어 텍스트 커서

③ 하드웨어 텍스트 커서

• 그래픽 커서

그래픽 커서는 그래픽모드에서만 나타나며 그 크기는 16도트×16도트로 256개의 도트(점)로 구성되며 일반적으로 화살표 모양을 하고 있다. 이 모양은 32바이트의 스크린 마스크와 32바이트의 커서 마스크로 정의되는데 사용자 임의대로 모양을 바꿀 수 있다.

• 핫 스팟(Hot Spot)

커서의 위치 결정을 하는 점으로 이 점을 중심으로 커서가 나타나게 된다. 256개의 점으로 구성된 마우스 커서가 어디를 그 위치점으로 할 지 결정을 지어주게 된다면 이해가 될 줄 안다. 가령 화살표의 경우는 꼭지점이, 손가락 모양의 경우는 손가락 끝이 핫 스팟이 될 수 있으며, 이 점 또한 임의로 변경할 수 있다.

• 소프트웨어 텍스트 커서

화면이 텍스트 상태에 있을 때 나타나는 커서로 그래픽 상태에서 처럼 다양한 모양을 이루지는 못하고 단지 문자의 속성을 조절하여 만드는 것이다. 그리고 그 모양은 ASCII코드의 256개 문자중 하나가 된다. 물론 모양은 사용자 임의대로 바꿀 수 있으며 그 정보는 16비트의 스크린 마스크와 16비트의 커서 마스크에 저장되어 진다. 마우스를 동작시키는 마우스 드라이버 소프트웨어는 우선 스크린 마스크와 해당 커서가 존재할 위치의 문자를 AND연산시키고 그 값을 커서 마스크와 XOR시켜 결과를 화면에 출력하는 원리를 갖는다.

그림 1은 문자의 구성을 보인 것으로 스크린 마스크와 커서 마스크도 이와 같이 구성시킨다면 다양한 모양을 얻을 수 있을 것이다. 한 예로

스크린 마스크 값=&H77FF

커서 마스크 값 =&H7700

으로 하면 직사각형의 반전된 사각형이 나타나게 된다. (뒤에 설명할 함수 10번을 통해 쓰이며 이 커서는 핫 스팟을 갖지 않는다.)

• 하드웨어 텍스트 커서

하드웨어 텍스트 커서는 컴퓨터가 켜진 후 DOS상태에서 나오는 커서이다.

(4) 버튼

마우스에는 제작하는 회사에 따라 2개 또는 3개의 버튼이 있다. 일반적인 조작을 보면 버튼을 누르지 않고 마우스를 움직이는 커서이동, 버튼을 재빨리 눌렀다 떼는 클릭, 버튼을 연속해서 2번 눌렀다 떼는 더블클릭, 그리고 버튼을 누르면서 마우스를 움직이는 드래그 등이 있다.

(5) 마우스의 이동 단위 미키

마우스가 이동해 다닐 때 그 움직임의 단위를 미키(Mickey)라 부른다. 즉 몇개의 미키가 기록되었나에 따라 몇개의 점을 이동시킬 것인가를 결정짓게 된다. 이것은 함수 15번을 통해 조절한다.

(6) 커서 플래그

마우스는 항상 커서의 표시 여부를 판단하기 위해 커서 플래그를 참조하게 되는데 그 값은 0이하의 값이 된다. 즉 커서 플래그가 0이면 커서는 화면에 나타나고 0이 아니면 (마이너스 수치) 커서는 사라지게 된다. 그러나 커서의 위치는 계속 기록된다.

함수 1번은 커서를 나타나게 하는 기능을 가지며 함수 2번은 커서를 사라지게 하는 기능을 갖고 있다. 함수 2번을 부를 때마다 커서 플래그는 하나씩 감소하게 되며 다시 커서를 보이게 하려면 커서 플래그가 0이 될 때까지 계속 함수 1번을 불러 주어야 한다. (이 플래그는 마우스 소프트웨어가 처리하는 것으로 사용자가 직접 플래그 값을 알려고 할 필요는 없다.)

(7) 마우스 사용을 위한 함수

마우스의 함수는 마우스 커서에 관한 것과 커서의 위치를 정하거나 알아보는 것, 버튼의 상태에 관계되는 것, 인터럽트와 관계되는 것 등 다양하게 준비되어 있다(표 1 참조).

함수를 설명하면서 보여주는 예제 프로그램은 누구나 이해하기 쉽도록 베이직으로 짰으며 앞에서도 잠깐 언급했듯이 자세한 기술적인 사항들은 앞글에 맡기기로 하고 여기서는 뒤에 소개하는 프로그램을 이해하는데 필요할 정도의 사항들에 대해서만 설명한다.

마우스 함수를 호출하기 위한 마우스정의는 prog#1에 보인 바와 같으며 이 루틴은 앞으로 설명되는 모든 예제 프로그램의 공통부분이 된다. 용법은

```
CALL MOUSE(A,B,C,D)
```

이다. prog#1을 보면 A에서 D까지를 정수로 정의하고 있는데 (DEFINT A-D) 이것은 마우스가 모든 변수를 정수로

〔그림 1〕 메모리내에서의 문자 구성

비트	15	14 12	11	10 8	7 0
	①	②	③	④	⑤

① 글자의 반짝임을 결정한다.
(1이면 반짝거리고 0이면 그대로 정지한다)
② 바탕색을 결정한다.
(허큘리스상에서는 무관하므로 111로 둔다)
③ 글자의 밝기를 조절한다.
④ 글자의 컬러를 조절한다.
⑤ 글자의 ASCII 코드값이다.

처리하고 있기 때문이다. 이와 같이 함수를 호출할 때 베이직에서는 A~D의 4개의 정수로써 이용하고 있는데 비해 C 언어에서는 a~e 5개를 이용한다. 가령 함수 1번을 호출하고 싶으면 베이직에서는

```
A=1
CALL MOUSE(A,B,C,D)
```

로 하면 되고 C에서는

```
a=1;
Mouse (&a,&b,&c,&d,&e);
```

와 같이 해야 하는 것이다. 그럼 실제로 함수에 대한 기본적인 사항들만 추려서 알아 보기로 하자. 여기서는 마이크로 마우스를 기준으로 기본적인 함수들만 선택해서 알아 보기로 하자.

• **함수 0번:** 초기화–prog4.0 참조

입력 : =0

출력 : A=마우스 상태

B=현재 마우스가 갖고 있는 버튼의 갯수

0번 함수는 마우스의 상태점검을 하고 초기화를 시켜 준다. 마우스가 없는 경우는 A의 값에 0이 되돌려지고 마우스가 있으면 −1이 되돌려진다. 한편 디폴트로 초기화 된 예를 보면 다음과 같다.

커서의 위치 : 화면 중앙

커서 플래그 : −1

그래픽 커서 : 화살표

텍스트 커서 : DOS상태의 커서

수평 Mickey / 점 비율 : 8대 8(1 : 1)

수직 Mickey / 점 비율 : 16대 8(2 : 1)

커서 이동범위 : 화면 전 영역

화면 번호 : 0

● **함수 1번:** 커서 나타내기–prog4.1 참조

입력 : A=1 출력 : 없음

함수 1번은 커서 플래그를 하나 증가시키고 그 값이 0이면 커서를 화면에 나타낸다. 일단 이 함수가 불리어지고 커서가 화면에 나타나면 커서는 마우스를 따라 같이 움직이게 된다.

• **함수 2번:** 커서 숨기기–prog4.2 참조

입력 : A=2 출력 : 없음

함수 2번은 커서를 지우고 커서 플래그를 하나 감소시킨다. 비록 커서는 없는 상태이지만 위치는 계속 추적중에 있게 된다. 이 함수는 화면에 커서가 차지하는 부분의 모드를 바꿀 때 이용되며 바꾸어진 화면을 유지하도록 한다. 그래서 화면의 상태가 바뀔 때(그래픽↔텍스트) 이 함수는 자동적으로 호출되며 다시 커서가 나타날 때 올바르게 나타나도록 한다. 또한 프로그램의 실행을 마칠 때도 이 함수를 호출해야 하는데 그렇지 않으면 커서가 화면에 계속 남아 있는 상태로 되기 때문이다.

〔표 1〕 마우스 함수번호와 기능

함수번호	기 능
0	마우스의 초기화
1	커서 보이기
2	커서 숨기기
3	버튼의 상태와 마우스의 위치 구하기(좌표)
4	마우스의 위치(좌표) 설정
5	버튼이 눌려진 횟수 구하기
6	버튼이 떼어진 횟수 구하기
7	최소, 최대 수평값 지정
8	최소, 최대 수직값 지정
9	그래픽 커서모양 정하기
10	텍스트 커서 지정
11	마우스 카운터의 읽기
12	특수상태에 따른 인터럽트 실행
13	라이트 펜 에뮬레이션 ON
14	라이트 펜 에뮬레이션 OFF
15	Mickey/Pixel 비율 지정
16	조건에 따른 커서 숨기기
19	속도 조절

〔prog #1〕

```
1000 '
1010 ' Mouse Driver Calling Routine
1020 '
1030 DEFINT A-D
1040 DEF SEG=0
1050 MSEG=256*PEEK(51*4+3)+PEEK(51*4+2)
1060 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)+2
1070 IF MSEG OR (MOUSE-2) THEN 1090
1080 PRINT"Mouse Driver not found":END
1090 DEF SEG=MSEG
1100 IF PEEK(MOUSE-2)=207 THEN 1080
1110 RETURN
```

• **함수 3번:** 버튼의 상태와 커서위치 구하기–prog4.3 참고.

입력 : A=3

출력 : B=버튼의 상태

C=수평 커서의 위치

D=수직 커서의 위치

3번 함수는 왼쪽, 오른쪽 그리고 가운데 버튼(없는 것도 있다)의 상태와 커서의 위치를 되돌려 준다. 버튼의 상태를

(prog 4.0)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
999 END
```

(prog 4.1)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
90 ' show cursor
100 A=1:CALL MOUSE(A,B,C,D)
999 END
```

(prog 4.2)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
90 ' show cursor
100 A=1:CALL MOUSE(A,B,C,D)
150 PRINT"press a key to hide cursor"
160 K$=INKEY$:IF K$="" THEN 160
190 ' hide cursor
200 A=2:CALL MOUSE(A,B,C,D)
999 END
```

나타내는 값은 비트조작을 해야 하는데 비트 0는 왼쪽 버튼을, 비트 1은 오른쪽 버튼을, 그리고 비트 2는 가운데 버튼을 나타낸다. 이 값이 0인 경우는 버튼이 올라간 상태이고 1이면 내려간 상태이다. 그리고 커서의 수평, 수직 위치는 화면에서 이동가능한 커서의 최소, 최대치 안에 있게 된다.

• **함수 4번:** 커서의 위치지정–prog4.4 참조

입력 : A=4
C=수평 커서 위치
D=수직 커서 위치
출력 : 없음

함수 4번은 C와 D의 값을 새로운 커서의 위치로 정하는데 사용된다.(단 그 값이 최대, 최소치를 벗어나면 안됨)

• **함수 5번:** 버튼 누르기에 관한 정보를 구함–prog4.5 참고.

입력 : A=5
B=버튼
출력 : A=버튼 상태
B=버튼이 눌려진 횟수
C=최후에 버튼이 눌려졌을 때 커서의 수평 위치
D=최후에 버튼이 눌려졌을 때 커서의 수직 위치

입력측에서 B는 체크할 버튼의 값을 나타내는데 이 값이 0이면 왼쪽 버튼, 1이면 오른쪽, 그리고 2이면 가운데 버튼을 의미하게 된다. 그리고 출력측의 A에는 버튼의 상태를 나타내는 비트가 기억되는데 비트 0에는 왼쪽 버튼, 비트 1에는 오른쪽 버튼, 그리고 비트 2번에는 가운데 버튼의 상태가 기억되며 그 방법을 보면 버튼이 올라간 상태이면 1이, 내려진 상태이면 0이 들어 있게 된다. 버튼이 눌려진 횟수는 B에서 받는데 1~32767사이의 값을 가진다.

• **함수 6번:** 버튼 떼기에 관한 정보 구하기–prog4.6 참조

입력 : A=6
B=버튼
출력 : A=버튼 상태
B=버튼이 떼어진 횟수
C=최후에 떼어졌을 때의 커서 수평위치
D=최후에 떼어졌을 때의 커서 수직위치

함수 6번은 5번과 동일하나 다만 몇번 눌려졌는가 아니면 몇번 떨어졌는가를 체크하는 차이 뿐이다.

• **함수 7번:** 수평의 최소, 최대 커서위치 지정–prog4.7 참조

입력 : A=7
C=최소 위치
D=최대 위치
출력 : 없음

함수 7번은 커서가 수평으로 움직일 때 그 범위를 제한하며 그 범위는 화면의 범위에 따라 달라지게 된다. 또 최소값이 최대값보다 크게 지정된 경우는 두 값이 서로 바뀌어지게 된다.

• **함수 8번:** 수직의 최소, 최대 커서위치 지정–prog4.8 참조

입력 : A=8
B=최소 위치
C=최대 위치
출력 : 없음

8번 함수는 7번과 동일하나 이번에는 수직위치와 수평위치의 제한이라는 차이점만 갖는다.

• **함수 9번:** 그래픽 커서지정–prog4.9 참조

입력 : A=9
B=수평 Hot Spot
C=수직 Hot Spot

(prog 4.3)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
90 ' show cursor
100 A=1:CALL MOUSE(A,B,C,D)
180 CLS
190 ' print button status and position
200 K$=INKEY$
210 A=3:CALL MOUSE(A,B,C,D)
220 LOCATE 1,1:PRINT USING "x ###";C
230 LOCATE 2,1:PRINT USING "y ###";D
240 LOCATE 3,1:PRINT"button is ";B
250 IF K$="" THEN 200
999 END
```

(prog 4.4)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
60 CLS
90 ' show cursor
100 A=1:CALL MOUSE(A,B,C,D)
190 ' set cursor position to 200,100
200 PRINT"press a key to set cursor position to 200,100"
210 K$=INKEY$:IF K$="" THEN 210
212 ' hide cursor
215 A=2:CALL MOUSE(A,B,C,D)
220 A=4:C=200:D=100
230 CALL MOUSE(A,B,C,D)
235 ' show cursor
240 A=1
250 CALL MOUSE(A,B,C,D)
999 END
```

(prog 4.5)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
60 CLS
90 ' get press information
100 A=1:CALL MOUSE(A,B,C,D) ' show cursor
110 PRINT"press a key, then the information will come
    out."
120 K$=INKEY$:IF K$="" THEN 120
130 A=5:B=0 ' left button status
140 CALL MOUSE(A,B,C,D)
150 PRINT"button status is ";A
160 PRINT"there was ";B;"button pressed"
170 PRINT"last position when button pressed was"
180 PRINT"x ";C;"  y ";D
999 END
```

(prog 4.6)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
60 CLS
90 ' get lift information
100 A=1:CALL MOUSE(A,B,C,D) ' show cursor
110 PRINT"press a key, then the information will come
    out."
120 K$=INKEY$:IF K$="" THEN 120
130 A=6:B=0 ' left button status
140 CALL MOUSE(A,B,C,D)
150 PRINT"button status is ";A
160 PRINT"there was ";B;"button lifted"
170 PRINT"last position when button pressed was"
180 PRINT"x ";C;"  y ";D
999 END
```

(prog 4.7)

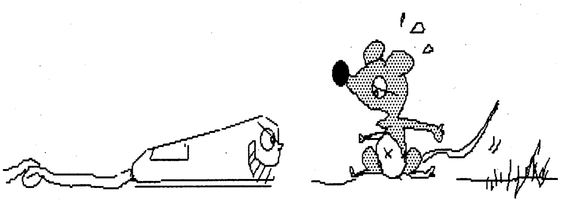
```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
60 CLS
90 ' set min X and max X of horizontal position
100 A=1:CALL MOUSE(A,B,C,D) ' show cursor
120 PRINT"press a key when ready"
130 K$=INKEY$:IF K$="" THEN 130
140 A=2:CALL MOUSE(A,B,C,D) ' hide cursor
150 ' set min to 100 and max to 300
160 A=7:C=100:D=300
170 CALL MOUSE(A,B,C,D)
180 A=1:CALL MOUSE(A,B,C,D) ' show cursor
999 END
```

(prog 4.8)

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
60 CLS
90 ' set min X and max X of vertical position
100 A=1:CALL MOUSE(A,B,C,D) ' show cursor
120 PRINT"press a key when ready"
130 K$=INKEY$:IF K$="" THEN 130
140 A=2:CALL MOUSE(A,B,C,D) ' hide cursor
150 ' set min to 50 and max to 150
160 A=8:C=100:D=300
170 CALL MOUSE(A,B,C,D)
180 A=1:CALL MOUSE(A,B,C,D) ' show cursor
999 END
```

(prog 4.9)

```
10 GOSUB 1000
60 A=0:CALL MOUSE(A,B,C,D)
80 DIM CURSOR(16,2)
100 CURSOR(0,0)=&HE1FF
110 CURSOR(1,0)=&HE1FF
120 CURSOR(2,0)=&HE1FF
130 CURSOR(3,0)=&HE1FF
140 CURSOR(4,0)=&HE1FF
150 CURSOR(5,0)=&HE000
160 CURSOR(6,0)=&HE000
170 CURSOR(7,0)=&HE000
180 CURSOR(8,0)=&H0
190 CURSOR(9,0)=&H0
200 CURSOR(10,0)=&H0
210 CURSOR(11,0)=&H0
220 CURSOR(12,0)=&H0
230 CURSOR(13,0)=&H0
240 CURSOR(14,0)=&H0
250 CURSOR(15,0)=&H0
300 CURSOR(0,1)=&H1E00
310 CURSOR(1,1)=&H1200
320 CURSOR(2,1)=&H1200
330 CURSOR(3,1)=&H1200
340 CURSOR(4,1)=&H1200
350 CURSOR(5,1)=&H13FF
360 CURSOR(6,1)=&H1249
370 CURSOR(7,1)=&H1249
380 CURSOR(8,1)=&HF249
390 CURSOR(9,1)=&H9001
400 CURSOR(10,1)=&H9001
410 CURSOR(11,1)=&H9001
420 CURSOR(12,1)=&H8001
430 CURSOR(13,1)=&H8001
440 CURSOR(14,1)=&H8001
450 CURSOR(15,1)=&HFFFF
500 A=9
510 B=5 : ' horizontal hot spot
520 C=0 : ' vertical hot spot
530 CALL MOUSE(A,B,C,CURSOR(0,0))
540 A=1:CALL MOUSE(A,B,C,D) ' show cursor
990 END
```



D=스크린 및 커서 마스크의 포인터

출력 : 없음

함수 9번은 커서의 모양, 색 그리고 커서의 중심을 지정한다.(이를 표시할 때는 함수 1번을 호출한다) 여기서는 스크린 마스크와 커서 마스크의 내용을 가지고 커서의 모양을 결정하는데 이 두 마스크의 내용을 마우스 루틴에 보내기 위해 그 값들을 배열에 넣은 후에 그 배열의 첫번째 내용을 D대신에 넣으면 된다. 예제 프로그램 및 뒤에 소개되는 드로잉 에디터 SEM_DRAW를 잘 살펴보기 바란다.

• **함수 10번 :** 텍스트 커서지정–prog4.10 참조

입력 : A=10

B=커서 선택

C=스크린 마스크값(스캔라인 시작 부분)

D=커서 마스크값(스캔라인 끝 부분)

출력 : 없음

함수 10번은 텍스트커서를 소프트웨어(B에 0을) 또는 하드웨어(B에 1을)적인 커서상태로 선택한다. 커서 설명란에서 언급했듯이 C와 D에는 각 마스크가 들어 가거나(소프트웨어 커서가 선택된 경우) 또는 스캔라인이 선택되어야 한다.(하드웨어 커서에서)

• **함수 15번 :** Mickey / 점 비율 지정–prog4.15 참조

입력 : A=15

C=수평 비율

D=수직 비율

출력 : 없음

함수 15는 미키 대 점의 비를 정한다. 즉 하나의 이동단위에 몇개의 점을 할당하느냐의 문제인데 1에서 32767사이의 값을 가질 수 있다. 보통 최초의 값은 수평으로 8이고 수직은 16이다.

실전 프로그램

(1) 프로그램 설명

이 프로그램(SEM–DRAW)은 마우스를 이용한 그래픽 에디터이다. 앞서 말했듯이 허큘리스 그래픽카드만 지원하며 그 외의 카드를 사용하고 있는 경우는 약간의 수정을 해야 한다. 그림 2에 전체 프로그램의 흐름도를 나타내었으니 참고하기 바란다.

(2) 함수 설명

• **void opengraph(void)**

이 함수는 그래픽모드를 지정하는 역할을 하며 지원 그래픽카드가 허큘리스인지를 판단하여 아니면 DOS로 빠져 나오는데 그래픽지원 카드가 허큘리스가 아닌 경우는 바로 이 부분을 수정하면 된다.(앞으로 수정할 부분이 많이 있는데 특히 수치 데이터들이 그렇다.)

• **gprint(int x, int y, char *fmt,…)**

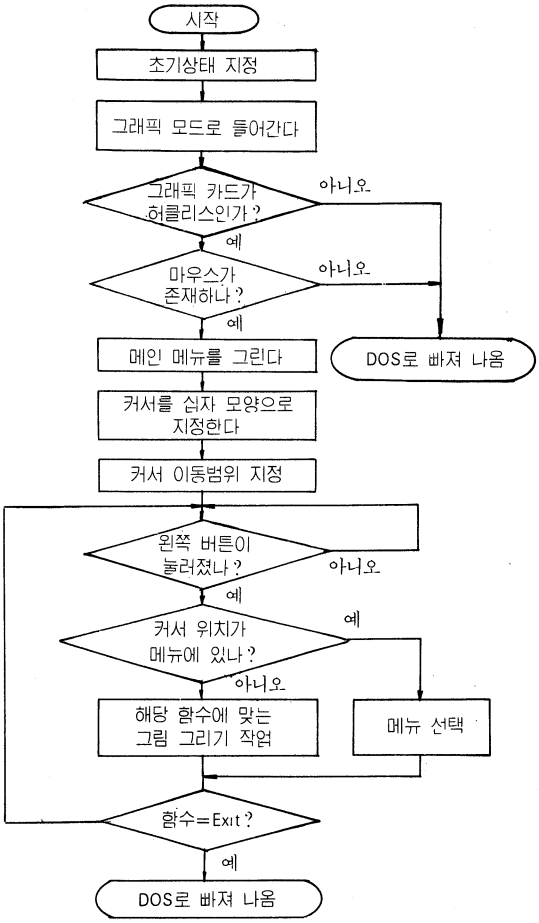
〔prog 4.10〕

```
1 SCREEN 0
10 GOSUB 1000
20 A=0:CALL MOUSE(A,B,C,D)
100 A=10
110 B=0 : 'select software text cursor
120 C=&HFFFF : ' screen mask
130 D=&H7700 : ' cursor mask
140 CALL MOUSE(A,B,C,D)
150 A=1:CALL MOUSE(A,B,C,D)
160 END
```

〔prog 4.15〕

```
10 GOSUB 1000
20 A=0
30 CALL MOUSE(A,B,C,D)
40 IF A=0 THEN PRINT"mouse is not installed":END
50 PRINT"mouse installed"
70 CLS
80 A=1:CALL MOUSE(A,B,C,D) ' show cursor
90 PRINT" press a key when ready"
100 K$=INKEY$:IF K$="" THEN 100
190 ' set mickey/pixel ratio to be slower
200 A=15
210 C=32:' default is 8
220 D=64:' default is 16
230 CALL MOUSE(A,B,C,D)
999 END
```

〔그림 2〕 SEM-DRAW의 흐름도



터보 C의 예제 프로그램 중에 BGIDEMO.C가 있는데 이 속에 있는 루틴을 수정하여 얻은 것이다. 우선 좌표값을 X와 Y에 넣고 그 이후는 printf와 같이 쓰면 되는데 이렇게 하면 그래픽 상태에서도 문자가 나온다.

터보 C에는 여러개의 문자폰트를 갖추고 있기 때문에 이들 문자들을 이용해도 되는데 단 하나의 문자크기가 바뀌면 SEM_DRAW 프로그램을 수정해야 하는 번거로움이 있다. 하지만 gprintf()에는 아무런 영향이 없다.

• point(int x, int y)

터보 C의 그래픽 함수를 보면 putpixel( )이 있는데 이 명령과 같은 기능을 갖는다. 여기서는 XOR기능을 첨가하여 허큘리스카드만 가능하게 하였는데 CGA 사용자는 Advanced Turbo C를 참조하기 바란다.

• int sgn(int a)

a값의 부호를 돌려주는 함수이다. Line()에서 이용된다.

• void Line(int sx, int sy, int ex, int ey)

비록 터보 C에 준비되어 있는 함수이기는 하지만 point()를 호출하기 위해 수정을 했다. Box()에서도 이용된다.

• void plot_circle(int x, int y, int xs, int ys)

원을 그리는 함수이다. 터보 C에 있는 루틴을 역시 point()를 이용하려고 새롭게 준비를 한 것이다.

• void Draw_box(int sx, int sy, int ex, int ey)

(sx, sy)에서 (ex, ey)를 대각선으로 하는 사각형을 그려준다.

• int OpenWindow(int xst, int yst, int xw, int yh)

이 함수는 xst와 yst를 좌측 상단으로 하여 가로로 xw, 세로로 yh의 윈도우가 열려진다. 현재 사용중인 문자체에 맞는 크기가 생성된다. 한번에 10개까지의 윈도우를 열수 있으며 컴파일을 COMPACT모드로 하면 윈도우의 크기를 제한없이 사용할 수 있다.

• int CloseWindow(void)

OpenWindow()에 의해 열려진 윈도우를 닫기 위해 호출한다. 이 함수에 의해 닫히는 윈도우는 가장 최근에 열려진 윈도우가 된다.

• void Mouse(int *m1, int *m2, int *m3, int *m4, int *m5)

이 함수가 실제 마우스 인터럽트를 호출하는 것으로 원래는 파스칼로 쓰여진 프로그램을 C로 수정을 하여 사용했다.

• void set_cursor(void)

터보 C에서 커서를 다르게 이용하고 싶을 때는 이 부분을 수정하면 된다.

• int check_mouse_install(void)

마우스가 현재 컴퓨터에 연결되어 있는지의 여부를 판단한다. 마우스를 찾지 못하면 DOS로 빠져 나온다.

• void system_set(void)

윈도우상에서 사각형의 커서를 만들어 내는 루틴이다.

• void draw_main(void)

본 프로그램의 메인 메뉴를 그리는 루틴이다.

• int select_drawing(void)

각각의 메뉴에서 선택을 하는 루틴으로 마우스의 왼쪽 버튼을 누르고 있는 동안은 계속 프로그램이 실행되는 곳이다.

• main()

본 프로그램의 메인 루틴이다.

(3) 프로그램 실행 및 사용법

리스트 1의 SEM_DRAW.C를 가지고 터보 C 1.5 이상의 버전에서 실행화일을 만들어 낸다. 마우스 드라이버 프로그램을 먼저 실행시키고 이 그래픽 에디터를 실행시킨다.

```
A:\>MSMOUSE
A:\>SEM_DRAW
```

〔리스트 1〕 SEM-DRAW.C

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <dos.h>

/*

  이 프로그램에서 사용되는 함수

*/
enum function {
        PENCIL = 0,
        LINE   = 1,
        BOX    = 2,
        FILL_BOX = 3,
        CIRCLE = 4,
        PAINT  = 5,
        ERASE  = 6,
        CLEAR  = 7,
        EXIT   = 8
};

char *menu[] = {
        " Pencil   ",
        " Line     ",
        " Box      ",
        " Fill Box ",
        " Circle   ",
        " Paint    ",
        " Erase    ",
        " Clear    ",
        " Exit     "
};
/*
 패턴형태 지정 Paint와 Fill-Box에 관여한다.

*/
char *patt[] = {
        " Empty     ",
        " Solid     ",
        " Line      ",
        " Ltslash   ",
        " Slash     ",
        " Bkslash   ",
        " Ltbkslash ",
        " Hatch     ",
        " Xhatch    ",
        " Interleave",
        " Wide Dot  ",
        " Close Dot "
};
/*

        컬러 타입

*/
char *m_color[] = {
        "  Black ",
        "  White "
};

void *menu_buffer=0; 메뉴 선택시 나타나는 흰색상자가
                     저장되는 곳

int    cursor[32];
int    drive_card=0;     /* 현재 쓰이는 그래픽 카드 종류 */
int    p_color=1;        /* 현재 쓰이는 색   */
int    put_type=0;       /* 점 찍는 형태 xor_put과 copy_put이 있다.*/
int    win_pointer=0;    /* 현재까지 화면에 있는 윈도우 갯수 */
double asp_ratio=0.0     /* 화면의 x와 y의 비율→원을 그릴 때 이용*/
```

```
void  *win_stack[10];      /* 윈도우의 화면이 저장되는 곳의 포인터 */
struct win_xy_buff {
        int xst;
        int yst;
        int xed;
        int yed;
} win_xy[10];

/*
    화면을 그래픽모드로 세트
*/
void opengraph(void)
{
    int mode=0;
    int xasp,yasp;

    drive_card=DETECT;
    initgraph(&drive_card,&mode,"");
    if (drive_card != HERCMONO) {      ← 허큘레스카드가 아닌 경우 DOS로 빠져 나옴
       cprintf("\n only use Hercules Graphics Card.\n");
       exit(0);
    }
    setcolor(WHITE);
    getaspectratio(&xasp,&yasp);
    asp_ratio = (double) yasp / (double) xasp;
}
/*
   그래픽 화면에 문자를 출력
   printf( )와 같이 쓴다.
   단, 앞에 x와 y의 위치가 들어간다.
*/
gprintf(int x, int y, char *fmt, ... )
{
   va_list   argptr;
   char str[140];
   int  cnt,high,length,width;
   int  xs,ys,xe,ye;
   int  fill=BLACK;

   va_start( argptr, format);
   cnt = vsprintf( str, fmt, argptr);

   high = textheight("H");
   length = textwidth(str);
   width= textwidth("W");
   setfillstyle(SOLID_FILL,fill);

   xs=(x-1)*width;
   ys=(y-1)*high;
   xe=xs+length;
   ye=ys+high;
   bar(xs,ys,xe,ye);
   setcolor(WHITE);
   outtextxy(xs,ys,str);
   va_end(argptr);
   return(cnt);
}
/*
   허큘리스의 지원으로 점을 찍는다.
   다른 그래픽 카드인 경우는 수정부분을 요구한다.
*/
point(int x, int y)
{
   char far *ptr = (char far *) 0xb0000000;
   int yadd;
   char first,last;

   if (x<0 || x>719 || y<10 || y>347) return;

   yadd = (y&0x3)*8192 + (y>>2)*90 + (x>>3);  ← 이 부분을 그래픽 카드에 맞게 수정한다.
   first = (char) *(ptr+yadd);
   last = 0x80>>(x & 0x07);
   switch (put_type) {
      case  COPY_PUT : *(ptr+yadd) = (p_color==0) ? first
                      & ~last : first | last;
                      break;

      case  XOR_PUT  : *(ptr+yadd) = first ^ last;
                      break;
   }
}
int sgn(int a)
{
   if (a==0) return 0  ;
   return (( a>0) ? 1 : -1);
}
/*
   ==========
   선 그리기
   ==========

   put_type 변수의 값에 따라 선의 성질이 달라진다.
   put_type이 xor_put이면 바탕색과 xor을  시키면서
   선이 그려지고 copy_put이면 바탕색과 관계없이
   현재 쓰이는 색(p_color)에 따라  선이 그려진다.
   xor_put과 copy_put는 graphics.h에 정의되어 있다.

*/
void Line(int sx, int sy, int ex, int ey)
{
   register int t, distance;
   int xerr=0, yerr=0, d_x, d_y;
   int incx, incy;
```

```
    d_x = ex-sx;
    d_y = ey-sy;

    incx = sgn(d_x);
    incy = sgn(d_y);
    d_x = abs(d_x);
    d_y = abs(d_y);

    if (incx==0) {
       distance=max(sy,ey);
       yerr=min(sy,ey);
       for (t=yerr;t<=distance;t++)
         point(sx,t);
    } else if (incy==0) {
       distance=max(sx,ex);
       yerr=min(sx,ex);
       for(t=yerr;t<=distance;t++)
         point(t,sy);
    } else {
       if (d_x>d_y) distance = d_x+1;
       else distance = d_y+1;

       for (t=0;t<=distance;t++) {
          point(sx,sy);
          xerr+=d_x;
          yerr+=d_y;
          if (xerr>distance) {
             xerr-=distance;
             sx+=incx;
          }
          if (yerr>distance) {
             yerr-=distance;
             sy+=incy;
          }
       }
    }
}
/*

    원을 그려주는 부함수

*/
void plot_circle(int x, int y, int xs, int ys)
{
   int x_start, x_end, x1, y_start, y_end, y1;

   y_start = y*asp_ratio;
   y_end = (y+1)*asp_ratio;
   x_start = x*asp_ratio;
   x_end = (x+1)*asp_ratio;

   for (x1=x_start;x1<x_end;x1++) {
      point(x1+xs,y+ys);
      point(x1+xs,ys-y);
      point(xs-x1,ys-y);
      point(xs-x1,y+ys);
   }
   for (y1=y_start;y1<y_end;y1++) {
      point(y1+xs,x+ys);
      point(y1+xs,ys-x);
      point(xs-y1,ys-x);
      point(xs-y1,x+ys);
   }
}
/*
    ===========
    상자 그리기
    ===========

    대각선으로 된 두 지점을 정의하면
    사각형이 그려진다.

*/
void Draw_box( int sx,int sy,int ex,int ey)
{
   Line(sx,sy,ex,sy);
   Line(ex,sy,ex,ey);
   Line(ex,ey,sx,ey);
   Line(sx,ey,sx,sy);
}
/*
    ===========
     원 그리기
    ===========

     xs와ys를 중심으로 원이 그려진다.
     rx와 ry는 원의 반지름을 구하는데 쓰이는 변수이다.

     반지름의 값은 (|rx-xs|+|ry-ys|)/2 로 된다.
*/
void Draw_circle( int xs, int ys, int rx, int ry)
{
   register int x,y,delta;
   int r;

   r=(abs(rx-xs) + abs(ry-ys)) / 2;
   y=r;
   delta = 3-2 * r;

   for(x=0;x<y;) {
      plot_circle(x,y,xs,ys);      ┐
      if (delta < 0)               │
         delta += 4*x+6;           │
      else {                       │
         delta += 4*(x-y)+10;      │
         y--;                      ├─ 원을 화면에 출력시키는
      }                            │  함수 호출
      x++;                         ┘
```

```
   }
   x=y;
   if (y)
      plot_circle(x,y,xs,ys);
}
/*
   ==========
   openwindow
   ==========
   그래픽 화면에 윈도우를 열어준다.
   왼쪽 상단(xst, yst)와 오른쪽 하단(xw, yh)를
   이 함수로  호출하면 윈도우가 나타난다.
   또 현재의 뷰포트는 윈도우 안으로 정의된다.
   한번에 모두  10개의 윈도우가 열려진다.

*/
int OpenWindow(int xst, int yst, int xw, int yh)
{
      int xloc, yloc, wloc, hloc;
      unsigned size;
      int txt_wth, txt_hgt;
      if (win_pointer == 9) return(-1);
      txt_wth = textwidth("W");
      txt_hgt = textheight("H");

      setlinestyle(SOLID_LINE,0,NORM_WIDTH);

      xloc = xst*txt_wth-2;
      yloc = yst*txt_hgt-2;
      wloc = (xw + xst )* txt_wth+2;
      hloc = (yh + yst )* txt_hgt+2;

      setviewport(0,0,719,347,0);

      win_xy[win_pointer].xst=xloc;
      win_xy[win_pointer].yst=yloc;
      win_xy[win_pointer].xed=wloc;
      win_xy[win_pointer].yed=hloc;

      size = imagesize(xloc, yloc, wloc, hloc);
      if (size == 0xffff) return(-1);
      win_stack[win_pointer] = malloc(size);
      getimage(xloc,yloc,wloc,hloc,win_stack
                                   [win_pointer++]);
      setfillstyle(SOLID_FILL,BLACK);
      bar3d(xloc ,yloc ,wloc ,hloc , 0, 0);
      setviewport(xloc +2, yloc +2, wloc - 2,hloc - 2,1);
}
/*
   ===========
   closewindow
   ===========
   OpenWindow에 의해 열려진 윈도우를 닫아준다.
   현재 열려진 , 윈도우가 없으면 그대로 돌아간다.

*/
int CloseWindow(void)
{
      int xloc, yloc;
      int hloc, wloc;
      if (win_pointer == 0) return(-1);
      xloc = win_xy[--win_pointer].xst;
      yloc = win_xy[win_pointer].yst;
      setviewport(0,0,719,347,0);
      putimage(xloc,yloc,win_stack[win_pointer],COPY_PUT);

      xloc = win_xy[--win_pointer].xst;
      yloc = win_xy[win_pointer].yst;
      wloc = win_xy[win_pointer].xed;
      hloc = win_xy[win_pointer++].yed;
      setviewport(xloc+2, yloc+2, wloc-2, hloc-2, 1);
      free(win_stack[win_pointer]);
}
/*
   ==============
     Mouse
   ==============
    이 함수가 마우스 드라이버를 호출한다.
    Mouse(8a, 8b, 8c, 8d, 8e); 로 부르면 된다.
    각 함수에 대해서는 설명란을 참고
*/
void Mouse( int *m1, int *m2, int *m3, int *m4, int *m5)
{
   union REGS r;
   struct SREGS s;

   if (*m1>=0) {
     r.x.ax = *m1;
     r.x.bx = *m2;
     r.x.cx = *m3;

      if ((*m1 == 9) || (*m1 == 12) || (*m1 == 20) ||
                      (*m1 == 22)  || (*m1 == 23)) {
         r.x.dx = FP_OFF(m4);
         s.es = FP_SEG(m4);
      }
      else if ( *m1 == 16) {
         r.x.cx = *m2;
         r.x.dx = *m3;
         r.x.si = *m4;
         r.x.di = *m5;
      }
      else
         r.x.dx = *m4;
```

```
    int86x(0x33, &r, &r ,&s);
    if (*m1 == 20)
      *m2 = s.es;
    *m1 = r.x.ax;
    *m2 = r.x.bx;
    *m3 = r.x.cx;
    *m4 = r.x.dx;
  }
}
/*
  그래픽 커서를 십자 모양으로 한다.
*/
void set_cursor(void)
{
  int a,b,c,d,e;

/*  스크린 마스크를 정의   */

  cursor[0] = 0xfc7f;   /* 1111110001111111 */
  cursor[1] = 0xfc7f;   /* 1111110001111111 */
  cursor[2] = 0xfc7f;   /* 1111110001111111 */
  cursor[3] = 0xfc7f;   /* 1111110001111111 */
  cursor[4] = 0xfc7f;   /* 1111110001111111 */
  cursor[5] = 0xfc7f;   /* 1111110001111111 */
  cursor[6] = 0x0001;   /* 0000000000000001 */
  cursor[7] = 0x0001;   /* 0000000000000001 */
  cursor[8] = 0x0001;   /* 0000000000000001 */
  cursor[9] = 0xfc7f;   /* 1111110001111111 */
  cursor[10] =0xfc7f;   /* 1111110001111111 */
  cursor[11] =0xfc7f;   /* 1111110001111111 */
  cursor[12] =0xfc7f;   /* 1111110001111111 */
  cursor[13] =0xfc7f;   /* 1111110001111111 */
  cursor[14] =0xfc7f;   /* 1111110001111111 */
  cursor[15] =0xffff;   /* 1111110001111111 */

/*  커서 마스크를 정의   */

  cursor[16] = 0x0000;  /* 0000000000000000 */
  cursor[17] = 0x0100;  /* 0000000100000000 */
  cursor[18] = 0x0100;  /* 0000000100000000 */
  cursor[19] = 0x0100;  /* 0000000100000000 */
  cursor[20] = 0x0100;  /* 0000000100000000 */
  cursor[21] = 0x0100;  /* 0000000100000000 */
  cursor[22] = 0x0100;  /* 0000000100000000 */
  cursor[23] = 0x7efc;  /* 0111111111111100 */
  cursor[24] = 0x0100;  /* 0000000100000000 */
  cursor[25] = 0x0100;  /* 0000000100000000 */
  cursor[26] = 0x0100;  /* 0000000100000000 */
  cursor[27] = 0x0100;  /* 0000000100000000 */
  cursor[28] = 0x0100;  /* 0000000100000000 */
  cursor[29] = 0x0100;  /* 0000000100000000 */
  cursor[30] = 0x0000;  /* 0000000000000000 */
  cursor[31] = 0x0000;  /* 0000000000000000 */
  a=9;b=7;c=7;      /* 핫 스팟을 7과 7.   */
                    /* 즉 십자의 교차점으로 정의한다. */
  Mouse(&a,&b,&c,&cursor[0],&e);
}
/*
  마우스 드라이버 프로그램이 시스템에 있는지의 여부를 판단한다.
  (없으면 DOS로 빠진다).
*/
int check_mouse_install(void)
{
  int a,b,c,d,e;

  a=0;          /*  함수 #0는 초기화를 실행한다. */
  Mouse(&a,&b,&c,&d,&e);

  if (!a) {
    closegraph();
    cprintf("\nInstall Mouse driver or plug in mouse.
     exit(0);                                   \n");
   }
   return 1;
}
/*
  메뉴에서 흰 사각형을 정의한다.
*/
void system_set(void)
{
   int size;

   setfillstyle(SOLID_FILL,WHITE);
   bar(0,0,119,7);
   size = imagesize(0,0,119,7);
   menu_buffer = malloc(size);
   getimage(0,0,119,7,menu_buffer);
   cleardevice();
}
/*
  메인 메뉴를 그린다.
*/
void Draw_Main(void)
{
   line(0,10,719,10);
   gprintf(1,1," Draw ");
   gprintf(25,1," Pattern");

   gprintf(11,1,"Pencil");
   gprintf(35,1,"Solid");

   gprintf(51,1,"Color");
   gprintf(61,1,"White");
}
/*
```

```
-----   SELECTING MODE   -----

   메뉴를 선택하는 부분이다. 왼쪽 버튼이 눌러진 동안은
   선택이 되며, 버튼을 떼면 해당 함수가 선택된다.
*/
int select_drawing(void)
{
   int a,b,c,d,e;
   int y=0,oy=0,ypos;
   int i;

   a=2;
   Mouse(&a,&b,&c,&d,&e);       /* 커서를 숨긴다. */

   OpenWindow(1,2,15,9);——— 윈도우를 열어준다.
   for (i=0;i<9;i++)
      gprintf(1,i+1,"%s",menu[i]);——— 메뉴 종류를 표시한다.

   a=3;            /* 함수 #3를 통해 버튼이 떼어짐을 체크한다. */
   Mouse(&a,&b,&c,&d,&e);
   ypos = y*8;
   putimage(0,ypos,menu_buffer,XOR_PUT);—[ 흰색의 사각형
                                           커서를 표시한다.

   while (b&1) {
      y=d/25;
      if (y>8) y=8;
      if ( y != oy ) {
         ypos = oy*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         ypos = y*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         oy=y;
      }
      a=3;
      Mouse(&a,&b,&c,&d,&e);
   }
   CloseWindow();
   gprintf(11,1,"%s",menu[y]);

   a=1;
   Mouse(&a,&b,&c,&d,&e);  /* ,  show mouse */

   return y;
}
int select_pattern(void)——[ 패턴을 선택하는 함수이다.
{                            패턴의 종류는 터보 C에서
                             제공되는 것을 그대로 이용한다.
   int a,b,c,d,e;
   int y=0,oy=0,ypos;
   int i;

   a=2;
   Mouse(&a,&b,&c,&d,&e);       /* 커서를 숨긴다. */
   OpenWindow(25,2,15,12);——[ 윈도우를 열어서 패턴 종류를
   for (i=0;i<12;i++)          화면에 표시한다.
      gprintf(1,i+1,"%s",patt[i]);

   a=3;
   Mouse(&a,&b,&c,&d,&e);
   ypos = y*8;
   putimage(0,ypos,menu_buffer,XOR_PUT);

   while (b&1) {
      y=d/25;
      if (y>11) y=11;
      if ( y != oy ) {
         ypos = oy*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         ypos = y*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         oy=y;
      }
      a=3;
      Mouse(&a,&b,&c,&d,&e);
   }
   CloseWindow();
   gprintf(35,1,"%s",patt[y]);

   a=1;
   Mouse(&a,&b,&c,&d,&e);       /* 커서 보이기    */

   return y;
}
void select_color(void)   /*   색을 선택하는 함수이다.    */
{
   int a,b,c,d,e;
   int y=0,oy=0,ypos;
   int i;

   a=2;
   Mouse(&a,&b,&c,&d,&e);       /* 커서 숨기기    */

   OpenWindow(50,2,15,2);
   for (i=0;i<2;i++)
      gprintf(1,i+1,"%s",m_color[i]);

   a=3;
   Mouse(&a,&b,&c,&d,&e);
   ypos = y*8;
   putimage(0,ypos,menu_buffer,XOR_PUT);

   while (b&1) {
      y=d/25;
      if (y>1) y=1;
      if ( y != oy ) {
```

```
        ypos = oy*8;
        putimage(0,ypos,menu_buffer,XOR_PUT);
        ypos = y*8;
        putimage(0,ypos,menu_buffer,XOR_PUT);
        oy=y;
      }
      a=3;
      Mouse(&a,&b,&c,&d,&e);
   }
   CloseWindow();
   gprintf(60,1,"%s",m_color[y]);

   a=1;
   Mouse(&a,&b,&c,&d,&e);      /*  커서 보이기  */

   p_color = y;
   setcolor(y);
}
/*
   MAIN FUNCTION
    which draws picture

*/
main()
{
   int a,b,c,d,e;
   int xe,ye,xs,ys;
   int fun=PENCIL;
   int patt=1;

   a=b=c=d=e=0;

   opengraph();
   pokeb(0x0040,0x0049,6);      /*  허큘리스에 커서가 나타나게
                                    표시한다. page #0은 6을,
                                    page #1은 5를 넣는다.  */
   check_mouse_install();←— 마우스가 있는지 체크
   system_set(); ———— 초기화

   Draw_Main(); ——— 메인 메뉴를 그린다.
   set_cursor();←—— 커서를 십자 모양으로 정의

   a=7;c=0;d=719;              /*  최대·최소 x의 변위를 지정 */
   Mouse(&a,&b,&c,&d,&e);
   a=8;c=0;d=347;
   Mouse(&a,&b,&c,&d,&e);  /*  최대·최소 y의 변위를 지정 */

   a=1;                        /*  show cu            */
   Mouse(&a,&b,&c,&d,&e);

   xe=0;ye=0;
   do {
      a=3;
      Mouse(&a,&b,&c,&d,&e);
      while (b&1) { ←— 왼쪽 버튼이 눌러진 동안 실행
         a=3;
         Mouse(&a,&b,&c,&d,&e);
         if (d<10) {   ←——— 메뉴의 글자 위에 커서가 있는가?
            if (c<150)                 /* draw 명령인가? */
               fun = select_drawing();
            else
            if (c>200 && c<280)        /*  패턴을 선택했나? */
               patt = select_pattern();
            else
            if (c>400 && c<480)        /* 컬러를 선택했나? */
               select_color();
         } else
         if (  (xe!=c || ye!=d || fun==PAINT || fun==CLEAR
                                                   || fun==ERASE)
 && d>10 && ye>10 ) {
 switch (fun) {              그림이 그려지는 부분에 커서가 있는지를 체크
   case PENCIL :
              a=2;
              Mouse(&a,&b,&c,&d,&e); /*  커서 숨기기 */
              setcolor(p_color);
              line(xe,ye,c,d);        /*  선 그리기  */
              a=1;
              Mouse(&a,&b,&c,&d,&e); /*  커서 보이기 */
              xe=c;
              ye=d;
              gprintf(70,1,"%5d",c); /*  화면에 현재의*/
              gprintf(80,1,"%5d",d); /* x, y축을 표시한다. */
              break;
   case  LINE :
              a=2;
              Mouse(&a,&b,&c,&d,&e);   /*  커서 숨기기    */
              xe=c;ye=d;
              xs=c;ys=d;
              put_type = XOR_PUT;
              Line(xs,ys,xe,ye);        /* 고무줄 성격의 선 그리기

                                        */
              a=3;
              Mouse(&a,&b,&c,&d,&e);
              while (b&1) {
                 a=3;
                 Mouse(&a,&b,&c,&d,&e);
                 if (xe!=c || ye!=d) {
                    Line(xs,ys,xe,ye); /* 고무줄을 지운다.   */
                    Line(xs,ys,c,d);   /*  선 그리기        */
                    xe=c;ye=d;
                    gprintf(70,1,"%5d",c);←——— x, y 표시
                    gprintf(80,1,"%5d",d);
                 }
              }
              Line(xs,ys,xe,ye);
```

```
        put_type = COPY_PUT;
        Line(xs,ys,c,d);
        a=1;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 보이기    */
        break;
case BOX :
case FILL_BOX : ←—— Box와 Fill Box는 같은 원리이다.
        a=2;
        Mouse(&a,&b,&c,&d,&e);      /* 커서를 숨긴다. */
        xe=c;ye=d;
        xs=c;ys=d;
        put_type = XOR_PUT;
        Draw_box(xs,ys,xe,ye);←— 처음의 고무줄 그리기

        a=3;
        Mouse(&a,&b,&c,&d,&e);
        while (b&1) {
           a=3;
           Mouse(&a,&b,&c,&d,&e);
           if (xe!=c || ye!=d) {
              Draw_box(xs,ys,xe,ye); /* 고무줄 지우기    */
              Draw_box(xs,ys,c,d);   /* 새로운 고무줄 그리기*/
              xe=c;ye=d;
              gprintf(70,1,"%5d",c);←— 화면 표시
              gprintf(80,1,"%5d",d);
           }
        }
        Draw_box(xs,ys,xe,ye);←— 고무줄 지우기
        put_type = COPY_PUT;
        if (fun == BOX)
           Draw_box(xs,ys,c,d);←— 상자 그리기
        else {
           setfillstyle(patt,p_color);
           setviewport(0,11,719,347,1);←┌ 패턴이 채워진
           bar(xs-11,ys,c-11,d);        └ 상자 그리기
           setviewport(0,0,719,347,1);
        }
        a=1;
        Mouse(&a,&b,&c,&d,&e);      /* 마우스 보이기 */
        break;
case  CIRCLE :
        a=2;
        Mouse(&a,&b,&c,&d,&e);          커서 숨기기
        xe=c;ye=d;
        xs=c;ys=d;
        put_type = XOR_PUT;
        Draw_circle(xs,ys,xe,ye);←— 고무줄로 된 원 그리기

        a=3;
        Mouse(&a,&b,&c,&d,&e);
        while (b&1) {
           a=3;
           Mouse(&a,&b,&c,&d,&e);
           if (xe!=c || ye!=d) {
              Draw_circle(xs,ys,xe,ye);←— 고무줄 원 지우기
              Draw_circle(xs,ys,c,d);←— 새로운 고무줄원 그리기
              xe=c;ye=d;
              gprintf(70,1,"%5d",c);
              gprintf(80,1,"%5d",d);
           }
        }
        Draw_circle(xs,ys,xe,ye);←— 고무줄원 지우기
        put_type = COPY_PUT;
        Draw_circle(xs,ys,c,d);←— 원 그리기
        a=1;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 보이기    */
        break;
case  PAINT :
        a=2;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 지우기    */
        setviewport(0,11,719,347,1);←┌ 메인 메뉴가
        setfillstyle(patt,p_color);  └ 지워지는 것을 방지
        floodfill(c,d-11,p_color);  /* 'd-11' is for */
                                    /*not painting main menu. */
        setviewport(0,0,719,347,1);
        a=1;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 보이기    */
        break;
case ERASE :
        setcolor(0);
        put_type = XOR_PUT;
        a=2;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 숨기기    */
        Draw_box(c-5,d-5,c+5,d+5);
        xs=c;ys=d;
        a=3;
        Mouse(&a,&b,&c,&d,&e);
        while (b&1) {
           a=3;
           Mouse(&a,&b,&c,&d,&e);
           if (xs!=c || ys!=d) {
              Draw_box(xs-5,ys-5,xs+5,ys+5);  ┌ 작은 사각형의
                          /* erase old box */ │ 영역을 검정색
                                              └ 으로 지운다.
              setfillstyle(SOLID_FILL,0);
              setviewport(0,11,719,347,1);
              bar(c-5,d-16,c+5,d-6);
                          /* erases in little box */
              setviewport(0,0,719,347,1);
              Draw_box(c-5,d-5,c+5,d+5);
                          /* draw new box */
              xs=c;ys=d;
              gprintf(70,1,"%5d",c);
              gprintf(80,1,"%5d",d);
           }
```

```
        }
        Draw_box(c-5,d-5,c+5,d+5);
        a=1;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 보이기   */
        setcolor(p_color);
        break;
case CLEAR :
        a=2;
        Mouse(&a,&b,&c,&d,&e);      /* 커서 숨기기   */
        setviewport(0,11,719,347,1);
        clearviewport();←— 메뉴는 그대로 두고 나머지 화면을
        setviewport(0,0,719,347,1);          모두 지운다.
        a=1;
        Mouse(&a,&b,&c,&d,&e);        커서 보이기
        break;
            }
        }
     }
     if (xe != c || ye!=d) {
        gprintf(70,1,"%5d",c);←— x와 y를 화면에 표시
            gprintf(80,1,"%5d",d);
            xe=c;ye=d;
        }
    } while (fun != EXIT);
    closegraph(); ←——          텍스트 모드로 돌아간다.
    printf("\n Bye. Bye. \n");
}
/*
   some part of this program refered to Bgidemo.c,
  the demo program in turbo c., and some parts are
  from 'Advanced Turbo C'.

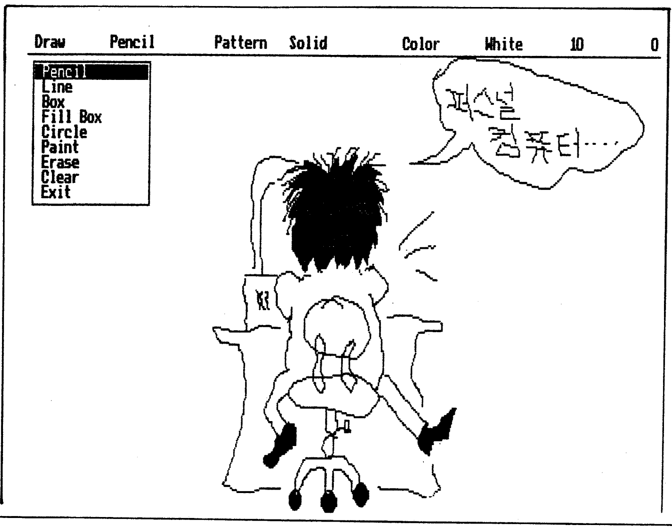
   This  program  is made only  for the purpose to
  demonstrate the usage of the Mouse.

   March 18. 1989.
                 Pragramed by Park Sae_hie.
                 Semtle. C.A.U.
*/
```

실행이 제대로 되면 작은 십자모양의 커서가 화면 중앙에 나올 것이다. 이 상태에서 마우스를 움직이면 커서도 같이 따라 움직이게 되며 디폴트로 선택된 메뉴는 연필과 흰 색상이 된다(화면 1 참조).

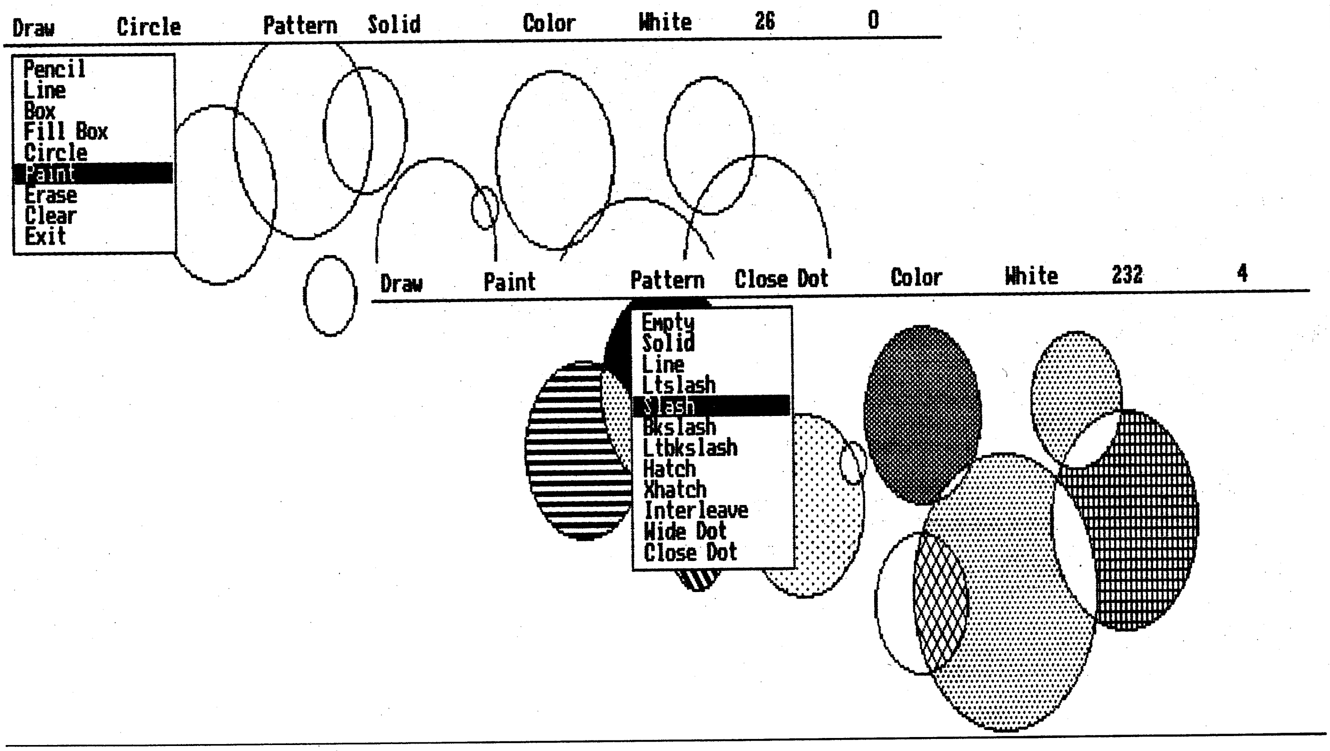
커서를 Draw나 Color, 또는 Pattern위에 놓고 마우스의 왼쪽 버튼을 누르면 해당 윈도우가 열리면서 선택메뉴가 나온다. 지금 버튼은 계속 내려진 상태에 있고 마우스를 밑으로 내리면 반전된 사각형의 커서도 밑으로 따라 내려오는데 이때 선택하고 싶은 위치를 놓고 버튼을 떼면 된다. 이렇게 하면 윈도우는 사라지고 선택상태가 된다.

〔화면 1〕 초기의 메뉴는 연필과 흰색상이 된다.



- **연필(Pencil):** 왼쪽 버튼을 누르고 있는 동안 선을 그려준다. 즉 연결되는 선이 십자 커서를 따라가며 그려지다가 버튼을 떼면 그리기를 마친다.
- **선(Line):** 마우스를 따라가며 가변길이의 선분이 그려진다. 선을 고정시키기에 적합한 위치를 정하면 버튼을 뗀다.
- **상자(Box):** 선과 유사하나 선 대신 상자 모양이 그려진다. 역시 크기와 위치를 마음대로 선택할 수 있다.
- **Fill Box:** Box와 같은 기능을 가지나 대신 Pattern으로 선택된 무늬가 채워진다.
- **Circle:** 상자 대신 원이 그려진다.
- **Paint:** 폐곡선으로 둘러싸인 영역을 선택된 무늬로 채운다(화면 2 참조). 그런데 가끔씩 원인모를 시스템 다운 현상이 일어나는데 필자가 미숙한지라 그 정확한 원인을 찾아내지를 못하고 다만 페인트하는 과정에서 스택이 모자라서 생기는 현상이 아닌가하고 추측을 해 보는데 실력있는 분의 조언을 부탁한다.
- **지우개(Erase):** 지우개를 선택하고 화면으로 내려와 왼쪽 버튼을 누르면 작은 사각형이 생긴다. 버튼을 누르면서 마우스를 이동시키면 그 사각형 밑은 모두 검은색상으로 변하게 된다. 즉 지우는 기능을 갖는다.
- **Clear:** Clear를 선택하면 화면의 어느 위치에서든지 왼쪽 버튼을 누름으로써 메인 메뉴를 제외한 모든 화면이 지워진다.
- **Exit:** DOS상태로 빠져 나온다.

〔화면 2〕 Paint와 Pattern 선택 예

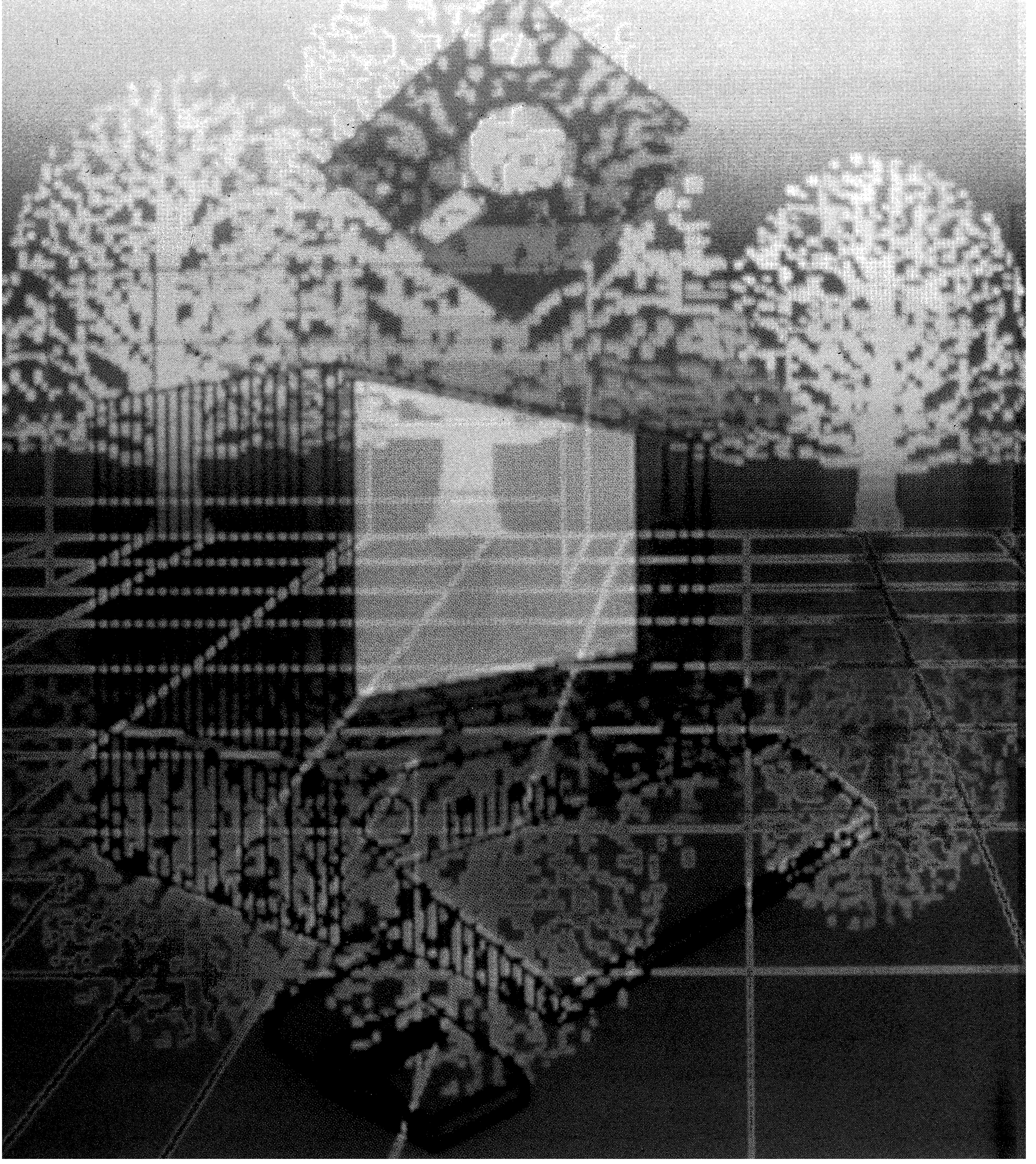


끝으로

마우스의 활용범위는 다양하다. 그래픽 분야에서 키보드는 이미 불편한 장비가 되어 버렸고 앞으로는 마우스를 비롯하여 라이트 펜, 스캐너 등의 전문 장비들이 인기를 모으고 있다.

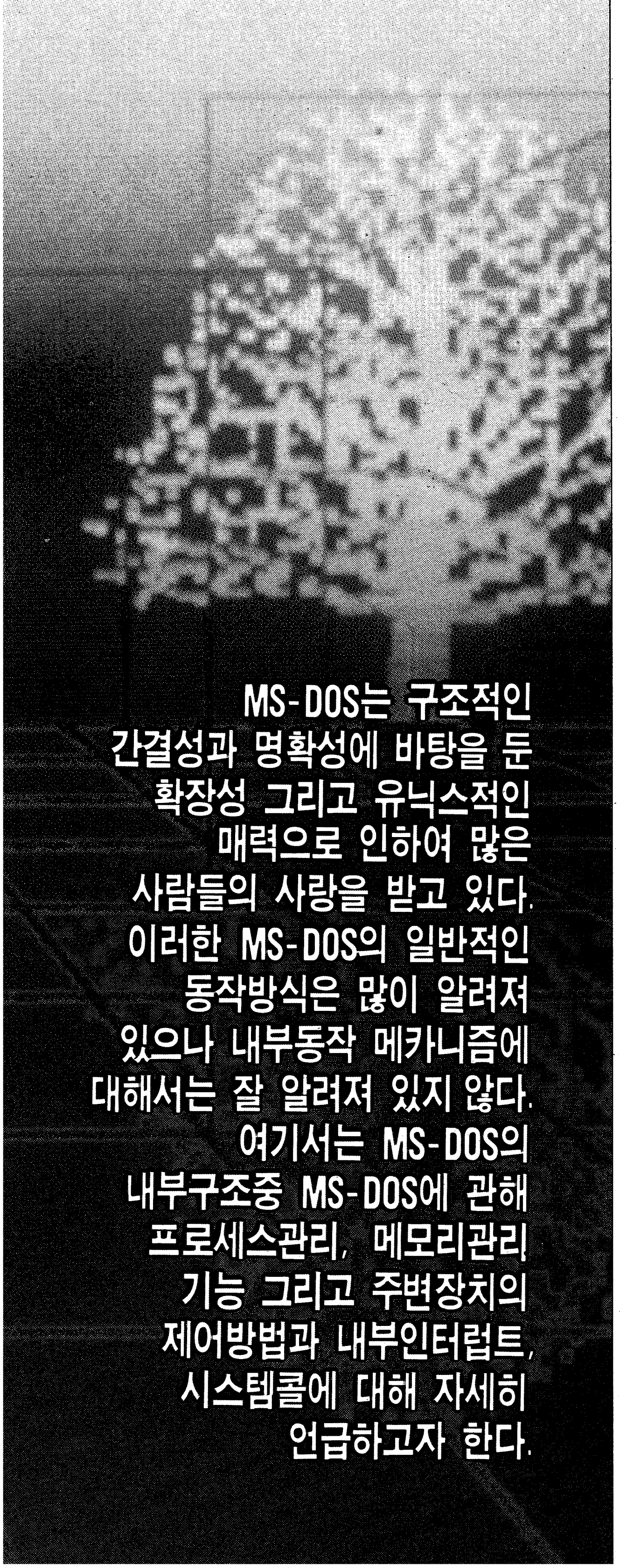
미숙하나마 마우스를 이용한 프로그램을 소개했는데 관심있는 독자들에게 어느 정도 도움이 되었다면 필자로서는 그 이상의 바램이 없겠다.

MSDOS.SYS의



분석

특집



MS-DOS는 구조적인 간결성과 명확성에 바탕을 둔 확장성 그리고 유닉스적인 매력으로 인하여 많은 사람들의 사랑을 받고 있다. 이러한 MS-DOS의 일반적인 동작방식은 많이 알려져 있으나 내부동작 메카니즘에 대해서는 잘 알려져 있지 않다. 여기서는 MS-DOS의 내부구조중 MS-DOS에 관해 프로세스관리, 메모리관리 기능 그리고 주변장치의 제어방법과 내부인터럽트, 시스템콜에 대해 자세히 언급하고자 한다.

홍근선/
아주대 전자과

IBM-PC와 함께 보급되기 시작한 MS-DOS는 나날이 발전하여, 사용자의 숫자나 사용기종의 종류 등 여러면에서 다른 DOS의 추종을 불허하는 DOS(Disk Operating System)의 대명사가 되었다. 이러한 이유는 IBM-PC의 세계적인 보급에도 힘입은 바 크지만, 그 밖에도 구조적인 간결성과 명확성에 바탕을 둔 확장성과 마이크로소프트사의 MS-DOS를 지원하는 풍부한 소프트웨어의 축적 등도 커다란 힘이 되었다.

또한 MS-DOS의 UNIX적인 매력과 하드웨어와 독립적인 구조의 명확성, 그리고 새로운 디바이스(기계적인 장치)사용에 대한 확장성은 16비트 PC 시장을 점령하기에 충분했고, 또 새롭게 멀티 태스킹(Multi-tasking : 여러 작업을 동시에 처리하는 기능) 기능을 추가한 MS-DOS V4.0의 개발로 32비트 PC시장에서의 확고한 기반을 확립하기 시작했다.

그런데 MS-DOS가 이렇게 폭넓게 자리잡고 있음에도 불구하고, 프로그래밍에 관심이 많은 대다수의 사용자가 그 기능과 동작을 충분히 이해하고 활용하지 못하고 있는게 사실이다. 본고에서는 이미 지난 3월호에 특집으로 게재된 MS-DOS의 구조에 이어서 본격적 활용을 위한 MS-DOS 내부동작의 메카니즘(특히 프로세스관리를 중심으로)과 이의 사용방법에 대해 살펴보고, "MSDOS.SYS"가 맡고 있는 기능에 대해서 중점적으로 언급하겠다.

"IO.SYS"가 맡고 있는 부분은 다음에 기회가 닿는데로 예제와 함께 다루기로 하고, 퍼스널컴퓨터에 이미 게재된 내용에 대해서는 중복설명을 피하였으나 어쩔수 없이 설명해야 하는 부분은 간단히 설명함을 원칙으로 하였다.

MS-DOS의 구성

그림 1에서 보는 바와 같이 중심의 하드웨어부(H/W)를 둘러싸고 있는 BIOS(Basic Input Output System)는 DOS를 하드웨어로부터 독립시키는 기능을 하는 부분으로 DOS로부터 입출력을 요구받아 하드웨어적으로 이를 실현한다.

BIOS부는 ROM BIOS와 DISK BIOS로 분리되는데, ROM BIOS는 하드웨어와 직접적으로 연관되는 부분으로 하드웨어 설계자 쪽에서 만들어져 시스템 ROM에 상주되고, 다른 DOS로부터 사용되어 진다.

반면에 DISK BIOS는 "IO.SYS"라는 이름으로 시스템 디스크에 들어있는데, 이 프로그램은 "MS-DOS. SYS"의 제어하에서 시스템과 접속되는 각

종 주변장치를 관리하기 위한 것이며, "디바이스 드라이버(Device Driver)"라는 다른 이름으로 불리운다.

MS-DOS의 강력한 기능중의 하나는 사용자가 손쉽게 새로운 장치, 예를 들어 플로터나 마우스같은 것들을 추가할 수 있다는 것이다. 사용자는 "CONFIG. SYS" 화일에 "DEVICE=" 명령을 사용하여 부팅시에 새로운 비상주 드라이버를 추가할 수 있다. 이러한 비상주 드라이버는 "IO.SYS"에 있는 상주 드라이버와 구별하기 위해 "설치가능한(Installable) 드라이버"라고 부른다.

이 설치가능한 드라이버는 다음 기회에 그 예와 함께 자세히 다룰 예정이므로, 관심있는 독자들의 많은 기대를 바란다.

실제로 MS-DOS는 "COMMAND.COM", "MSDOS. SYS", "IO.SYS"의 세개의 화일로 구성되어 있고, 이중 "MSDOS.SYS"와 "IO.SYS"는 코맨드 레벨의 일반 사용자 실수로 파괴될 것을 염려하여 숨은 화일(Hidden File)로 되어 있다.

"COMMAND.COM"은 일반 사용자가 입력한 명령을 해석하여 이를 실행하는 기능을 담당하는 부분으로, 그 일반적인 동작방식이 많이 알려져 있고, 또 지난 3월호에서 비교적 상세히 언급되었으므로 여기서는 생략하기로 한다. 그리고 "MSDOS. SYS"는 MS-DOS의 특성을 결정짓는 중심부분으로, 주기능은 화일관리, 메모리관리, 프로세스관리 그리고 주변장치의 제어를 담당하고 있다.

본고에서는 동적인 프로그램의 설계를 위해 "MSDOS. SYS"의 기능중 지난 호에서 지면상 언급이 미약한 프로세스관리 및 메모리 관리기능에 대해서 중점적으로 알아보기로 한다.

먼저 MS-DOS상에서 프로그래밍시 "MSDOS. SYS"의 기능을 빌어, 자주 이용하는 내부인터럽트와 시스템콜에 대해서 언급하겠다.

내부 인터럽트의 기능

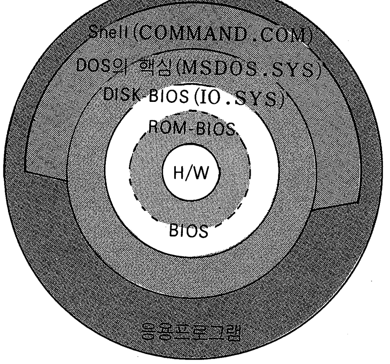
"MSDOS.SYS"의 여러 기능들은 DOS의 서비스루틴으로 사용자에게 개방되어 있고, 서비스루틴과 함께 사용자 접속방식으로 8086 CPU의 소프트웨어 인터럽트를 사용하고 있다.

8086 CPU에서는 사용자 정의 소프트웨어 인터럽트로서 20H-FFH까지 사용할 수 있는데, 이 중 MS-DOS가 사용하는 것은 20H에서 3FH까지의 32개이다. 여기서 일반 사용자가 이용할 수 있는 것은 20H~27H인데, 이 중 INT 22H~24H는 MS-DOS가 어떤 필요에 의해서(뒤에서 설명된다) 단지 어드레스를 놓는데에 사용하고 있기 때문에 이를 명령으로 사용할 수가 없다. 그리고, 나머지 내부인터럽트는 마음놓고(?) 응용 프로그램 제작시 사용하여 DOS가 제공하는 모든 기능을 서비스 받을 수 있다(표 1 참조).

(1) INT 20H와 INT 27H

표 1에서 보는 바와 같이 INT 20H와 INT 27H는 모두 프로그램을 종료시키는 것으로 사용하는

〔그림 1〕 MS-DOS의 계층구조



〔표 1〕 내부 인터럽트 일람표

인터럽트 번호	기 능
INT 20H	프로그램의 종료(비상주 종료) CS레지스터는 PSP의 패러그래프 번호가 들어 있어야 한다.
INT 21H	프로그램의 종료 어드레스
INT 22H	프로그램의 종료 어드레스
INT 23H	CTRL-C 어드레스
INT 24H	치명적 에러 처리 루틴의 어드레스
INT 25H	물리적 섹터 번호에 의한 디스크 읽기 콜 ┌AL←드라이브 번호 │DS : BX←DAT │CX←읽기 섹터의 수 └DX←읽기 시작하는 상대 섹터 리턴┌CF=0 정상 종료 └CF=1 에러 발생(AL레지스터에 에러 코드를 되돌림)
INT 26H	물리 섹터 번호에 의한 디스크 써넣기 콜 ┌AL←드라이브 번호 │DS : BX←DTA │CX←기입 섹터의 수 └DX←기입 시작하는 상대 섹터 리턴┌CF=0 정상 종료 └CF=1 에러 발생(AL레지스터에 에러 코드를 되돌림)
INT 27H	프로그램의 상주 종료 CS : DX←상주부의 최종 바이트 다음에 오는 최초의 바이트 어드레스

데, 그 차이는 INT 20H는 프로그램이 점유했던 메모리가 해체될 수 있다는 것을 운영체계에 알리고, INT 27H는 그 프로그램을 메모리에 상주시켜 계속 유지할 수 있게 해주는데에 있다.

그리고 이 INT 20H와 INT 27H는 각각 도스 기능의 00H, 4CH와 31H로 대체될 수 있다.

이 두개의 내부인터럽트를 사용하기 전에 주의해야할 것은 오픈된 화일은 데이터가 파괴될 수 있기 때문에 반드시 닫아 주어야 한다. 이의 자세한 동작의 예는 프로그램 실행과 종료에서 다루겠다.

(2) INT 21H

내부인터럽트의 21H는 시스템콜을 위해 사용되고 있는데, 도스 버전 3.XX에는 사용자에게 공개된 기능이 예약된 것까지 무려 100여개나 되어 필요에 따라 다양한 기능을 제공받을 수 있게 되었다. 이 INT 21H는 본고의 마지막부분에서 그 종류에 대해서만 알아보기로 하고, 더 이상의 자세한 설명은 지면 관계상 생략하기로 한다. 참고로 시중에 나온 책중에 INT 21H의 종류와 기능들에 대해 자세히 서술된 책이 많으므로 이를 참조하면 도움이 되리라 생각된다.

INT 21H가 프로그램 내부에서 발생했을 경우, 그 프로그램의 제어권은 "MSDOS.SYS"의 시스템콜 처리루틴으로 옮겨져 여기서 AH레지스터에 들어 있는 기능번호가 올바른지를 조사하고, 그 번호가 올바른 경우, 각 레지스터를 대피시켜 놓은 다음에 기능번호에 따른 처리루틴으로 분기하여(물론, 그 기능번호가 바르지 않을 경우에는 그대로 프로그램으로 되돌아 간다.) 수행하고, 필요한 경우는 그 결과를 특정 레지스터에 정리한 후에 IRET를 만나 원래의 프로그램으로 되돌아 간다. "MSDOS.SYS"의 기본적 처리는 이상과 같지만, 실제는 처리방법이 무척 복잡하게 되어 있다.

예를 들어, 〈CTRL-C〉가 눌러졌는지를 판단하여 필요에 따라 INT 23H의 처리루틴을 호출하기도 하고, 만약 이 루틴이 불리어질 경우에는 다른 시스템콜이 불려질 경우를 대비해 "MSDOS.SYS"의 작업영역과 스택이 깨지지 않도록 처리한다.

(3) INT 22H, 23H, 24H

이 3가지 내부인터럽트는 프로그램을 종료할 때나 중단할 때, 현재 프로그램의 제어권을 받게 되는 루틴의 주소를 가지고 있다. 단, 이들은 프로세스 기동시 인터럽트 벡터로부터 PSP에 복사되어 종료 또는 중단시 다시 원래의 자리로 되돌아갈 수 있으므로 독자적인 처리루틴을 가지고 있다. 프로그램의 실행과 종료에서 이를 자세히 설명하겠다.

특집

(4) INT 25H, 26H

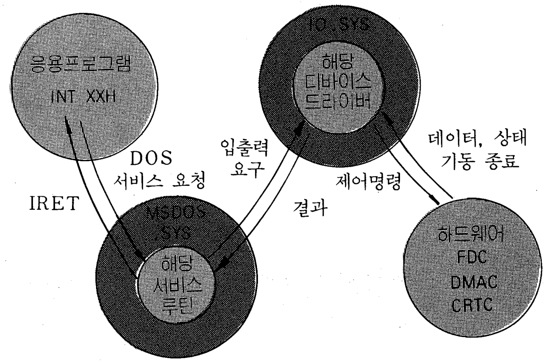
이들 2개의 내부인터럽트를 이용하여 디스크상의 화일과는 관계없이 임의의 섹터에 읽기와 쓰기를 할 수 있다. 이 기능은 DISKCOPY 등에서 유용하게 사용된다(그림 2 참조).

프로그램의 실행과 종료

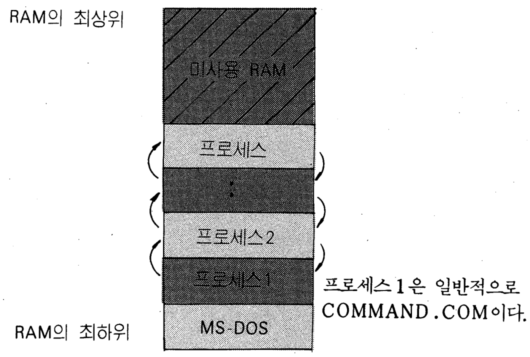
MS-DOS에서는 메모리중에 적재되어 실행되는 프로그램을 "프로세스(Process)"라 하고, 프로세스의 기동과 종료 그리고 이에 따른 관리를 "프로세스 관리"라 한다.

MS-DOS 버전 3.XX는 싱글 태스트, 싱글 유저를 위한 DOS인데, 그림 3의 복수 프로세스의 기동과 같이 복수개의 프로세스를 기동시킬 수 있다.

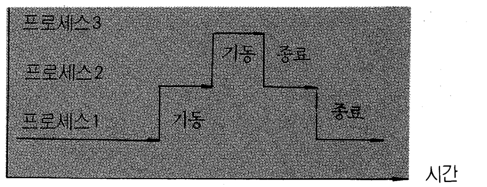
〔그림 2〕 내부 인터럽트에 의한 기능 표출



〔그림 3〕 복수프로세스의 기동



〔그림 4〕 복수프로세스의 수행



단, 프로세스들이 동시에 수행되는 것이 아니고, 그림 4의 복수 프로세스의 수행과 같이 자프로세스를 기동한 모프로세스는 자프로세스의 실행이 끝날 때까지 기다려야 한다(버전 4.0에서는 이러한 제약 없이 멀티 태스킹이 가능하다).

여기서 모프로세스와 자프로세스는 어떤 프로세스가 수행중에 새로운 프로그램을 적재하여 실행시키는 경우에는 원래의 프로그램을 "모프로세스(Parent Process)" 혹은 "친 프로세스"라 하고 새로이 기동된 프로세스를 "자프로세스(Child Process)" 라 부른다.

실행 가능한 프로그램은 2가지 형태로 존재하는데, 이들은 각기 서로 다른 구조와 기동방식을 사용한다. 여기서는 이들 2가지 형태의 실행화일 구조와 기동방식, 그리고 DOS와 수행될 프로세스간의 대화용 영역인 PSP(Program Segment Prefix)에 관해 알아보고, 프로세스의 실제적인 기동 예와 프로세스 실행환경의 관리기법 및 프로세스 종료방식을 언급하겠다.

(1) 실행형 화일의 구조

MS-DOS하에서는 바로 위에서 언급한 것처럼 실행 가능한 프로그램으로 2가지 형태가 있는데, 하나는 최대길이가 64K바이트로 제한되는 COM모델이고, 다른 하나는 가능한 모든 메모리를 사용할 수 있는 EXE모델이 있다.

이들은 각각 외부기억장치인 디스크상에 서로 다른 모양으로 존재하는데, COM모델(확장명이 COM)의 프로그램은 절대적인 코드, 즉 순수한 기계어 코드들로만 디스크상에 저장되어 있고, EXE모델(확장명이 EXE)의 프로그램은 그 화일에 관한 정보인 헤더와 재배치 정보, 체크썸 등 MS-DOS에 의해서 사용될 정보를 가지고 있는 모습으로 디스크에 존재한다.

이렇게 서로 다른 모습으로 디스크상에 존재하더라도 COM모델과 EXE모델의 프로그램은 둘다 비슷한 메카니즘으로 읽혀지고 수행되는데, MS-DOS는 실행될 프로그램이 놓일 메모리를 할당하고, 그곳에 프로그램 세그먼트 프리픽스(PSP)를 만들고 나서, PSP 상단의 기억장소에 프로그램을 읽어 넣는다.

그리고 MS-DOS(EXEC기능)는 세그먼트 레지스터와 스택을 설정하고 제어를 프로그램에 넘겨준 후, 각종 정보를 그 프로그램에 제공해준다.

하나의 실행형 화일이 수행되는 기본적 절차는 이상과 같은데, 여기서는 각 실행형 화일의 구조와 기동의 메카니즘을 구체적으로 알아보자.

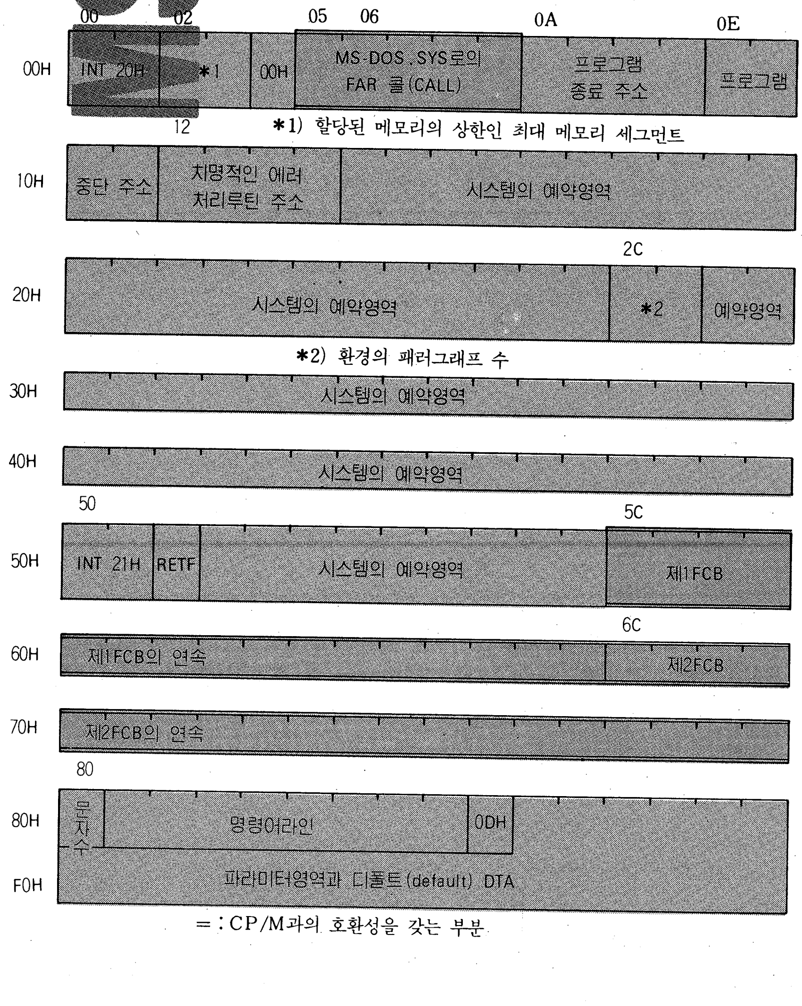
●PSP

프로그램이 메모리상에 로드되어 실행될 때, 그 프로그램이 놓이게 되는 곳의 선두 256바이트에는 "MSDOS.SYS"가 "COMMAND.COM"에 의해서 건네지는 파라미터의 문자열이나 각종 정보들을 저장하는데 이를 PSP라 부른다. 이 PSP는 다음 절에서 설명하게 되는 종료나 중단 어드레스, FCB, DTA, "MSDOS.SYS"의 작업영역이 포함되어 있다(그림 5 참조).

MS-DOS 프로그래머이라면, PSP에 대해 완전히 이해하고 있는 것이 MS-DOS하에서의 성공적인 프로그래밍을 위한 필수적인 것이라 할 수 있다.

PSP의 구성은 그림 5와 같이 되어 있으며, 그 각각의 내용을 살펴보면 아래와 같다(리스트 1 참조).

〔그림 5〕PSP의 구체적인 구조



= : CP/M과의 호환성을 갖는 부분

00H~01H

여기에는 INT 20H의 코드가 들어 있는데, 이는 프로그램의 작업을 끝내고, 메모리의 해방을 수행하는 프로세스종료 핸들러이므로, 프로그램에서 PSP의 00H로 점프하는 동작으로 프로그램을 종료할 수 있게 된다.

02H~03H

PSP의 오프셋 02H에서부터 03H까지는 프로그램에 할당된 메모리블록의 상위 세그먼트 어드레스를 저장하고 있는데, 이는 그 프로그램의 작업을 수행하는데, 더 많은 메모리를 요구하든지 남은 기억장소를 다른 프로세스에 넘기는데 사용하고 있다(그림 6 참조).

05H~09H

이곳에는 "MSDOS.SYS"에 의해 FAR CALL 명령이 들어 있다. 일반적으로 시스템콜의 기본적 사용방법은 AH레지스터에 펑션번호를 부여하고 INT 21H를 수행하는 것인데, CP/M 80과의 호환성을 위해 CL레지스터에 펑션번호를 넣고 PSP의 05H를 CALL하여 같은 결과를 얻을 수 있다.

그러나, 이 방법을 사용하면, AX레지스터는 항상 파괴되어 AL레지스터에 파라미터나 그 펑션의 결과를 리턴하지 않으므로, CP/M80과 호환성을 고려하는 일부 기능에만 사용하고 있다.

또, 프로그램이 EXE모델일 경우 스택을 조정해야 하므로 CP/M과의 호환성을 고려하지 않는다면 COM모델을 제외하고는 사용하지 않는 것이 바람직하다(그림 7 참조).

0AH~15H

여기에는 3개의 더블워드 포인터가 존재한다. 이들은 PSP의 작성시에 INT 22H~INT 24H의 인터럽트 벡터들이 복사된 것이며, 프로그램에서 그 벡터를 수정하여 독자적인 〈CTRL-C〉와 치명적 에러 처리루틴을 가질 수 있게 하고, 프로그램 종료시에 이곳의 정보가 다시 INT 22H~24H 벡터에 되돌려진다.

자세한 예는 프로세스의 기동과 종료에서 상세히 다루겠다.

2CH~2DH

여기에는 환경블록의 세그먼트 어드레스가 들어있는데, 이 환경블록은 수행될 프로그램의 환경정보들로 구성되고, 이러한 정보들은 일반적으로 이전의 프로세스 환경의 본보기이다.

이 환경정보에는 실행할 프로그램을 발견하기 위해 "MSDOS.SYS"에서 사용될 탐색경로나 "COMMAND. COM"의 디스크상의 위치, "COMMAND.COM"에 의해 시용될 프롬프트의 형태에 관한 것들이 들어가 있다.

50H~52H

PSP의 오프셋 50H부터 3바이트는 INT 21H, RETF 명령어가 들어 있어, 위의 05H~09H와 그 기능은 같은데, 위의 경우처럼 제한적이지 않아 INT 21 다음에 시스템콜을 하기 위해 이를 사용할 수 있다. 즉, 시스템콜을 위해 PSP의 50H로 CALL 해도 좋다.

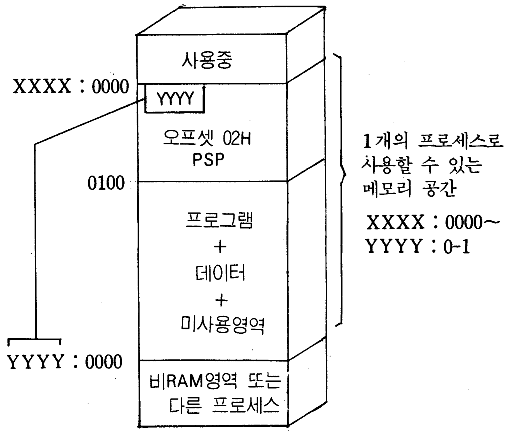
5CH~6CH

이 5CH와 6CH에는 FCB(화일 콘트롤 블록)이 있다. FCB는 화일에 대한 오픈, 클로즈, 읽기, 쓰기, 삭제동작에 사용되는 작업영역이라 할 수 있는데, 여기에 대한 설명은 후에 화일핸들과 함께 한다.

80H~0FFH

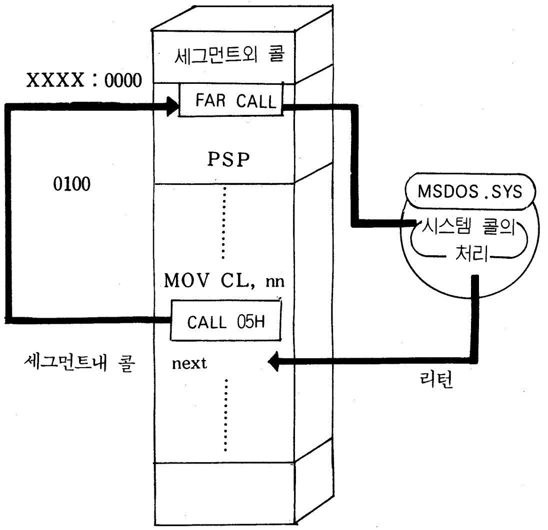
여기서 00FFH번지까지의 128바이트는 수행될

〔그림 6〕 PSP의 사용 가능한 바이트 수



PSP의 오프셋 02H의 값(이 경우는 YYYY)은 최초에 PSP가 만들어졌을 때에 세트되면 다음은 그대로의 값을 유지하고 있고, 메모리 관리용 시스템 콜에서 메모리영역의 크기를 변경해도 이 값은 변하지 않는다.

〔그림 7〕 CALL 5에 의한 시스템콜



프로그램으로 데이터 전송을 위해 사용된다. 일반적으로 명령어 꼬리부분이 이곳에 저장된다.

●COM형 화일의 구조와 기동

앞에서 어느 정도 언급했지만 COM형 화일은 EXE 형 화일과 달리 기계어로만 디스크상에 저장되어 있어 동일한 기능을 하는 EXE형 화일보다 크기가 작으며, 실행을 위한 메모리에 적재하는 시간도 큰 차이는 없지만 약간은 빠르다.

또한 "MSDOS.SYS"는 그 화일이 실행 가능한지 판단하는 정보가 COM형 화일에는 존재하지 않기 때문(EXE형 화일에는 오버플로우를 무시하여 체크썸한 값을 정보로 가지고 있다.)에 확장명이 COM인 화일에 대해서는 그대로 메모리에 옮겨 각 코드들을 해석하여 실행한다.

이렇게 COM형 화일의 구조는 말할 것도 없이 (이미 위에서 설명했지만) 무척 간단하다. 그리고 "MSDOS.SYS"에 의해 COM형 화일이 실행하게 되는 메카니즘을 알아보면 아래와 같다.

먼저 메모리를 0FFFFH만큼 확보한 뒤 PSP를 작성하고, 프로그램을 PSP 바로 뒤에 적재한다. 그런 다음 제어권을 프로그램에 넘겨주기 전에 모든 세그먼트 레지스터들을 PSP의 세그먼트로 하고, IP레지스터를 100H로 하여 실행하는 것이다.

●EXE형 화일의 구조와 기동

EXE형 화일은 **그림 8** EXE형 화일의 구조에서 보는 것과 같이 헤더부와 로드 모듈부로 구성되어 디스크상에 존재하는데, 이 헤더에는 EXE형 화일의 실행을 위한 제어정보가 들어가 있다.

"MSDOS.SYS"는 이 헤더 중에서 프로그램을 메모리에 재배치하기 위한 정보를 읽어 로드 모듈을 재배치한다.

헤더의 구성은 **표 2**에 나타나 있다.

그러면 위에서 언급한 헤더에 따라 어떻게 EXE 형 화일이 메모리에 적재되어 실행되는지 알아보자(**리스트 2** 참조).

"MSDOS.SYS"는 제일 먼저 프로그램을 실행하는데 필요한 메모리블록을 확보한다. 다시 말해서 메모리 중에서 현재 사용할 수 있는 영역을 확보하고, 프로그램 실행시 필요한 최소한의 메모리(헤더의 0AH~0BH)와 패러그래프 단위로 표시된 로드 모듈의 크기를 더한 값이 확보한 메모리 영역의 크기보다 크면 에러로 처리하고, 코맨드 라인 상태로 돌아간다.

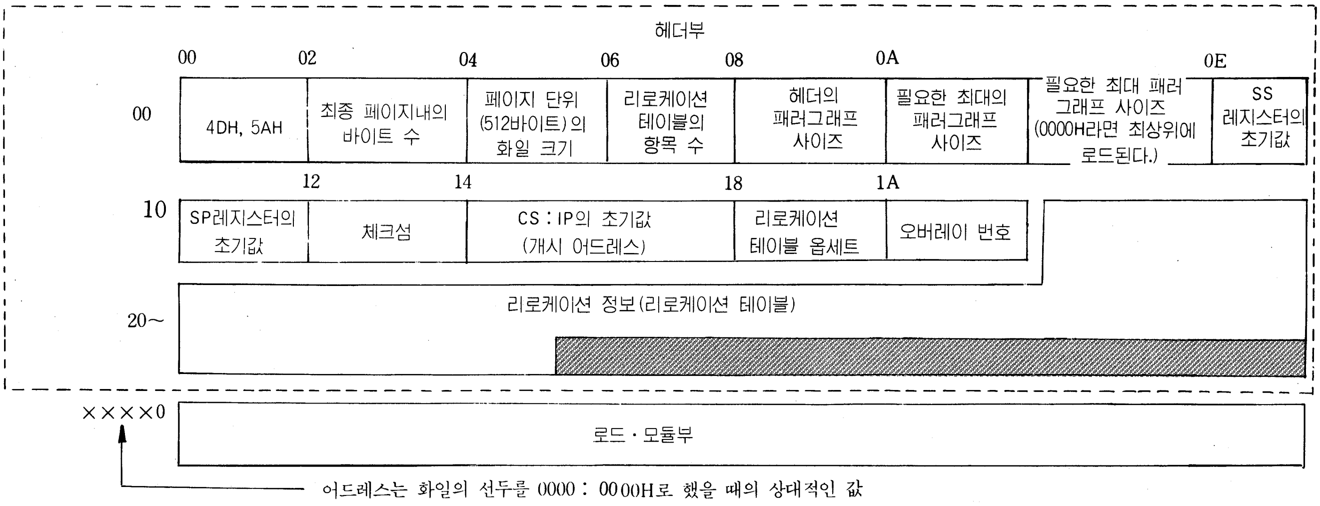
여기서 실행시 최소한의 필요한 메모리라는 것은 프로그램 수행중에 사용할 최소한의 메모리 크기를 뜻한다. 이외에 메모리 확보를 위한 처리는 몇가지가 더 있으며, 이들은 프로그램의 실행을 위해서 필요한 영역 확보를 위해서 필요한데 여기서는 생략하기로 한다.

이제 프로그램 실행을 위해 필요한 메모리는 확보되었기 때문에 "MSDOS.SYS"는 PSP를 작성하고, 프로그램의 로드 모듈을 PSP 바로 뒤에 적재하게 된다.

그리고, 바로 각 모듈들의 재배치에 들어가는데 헤더의 재배치 정보에 따라 각 로드 모듈의 세그먼트 정보를 계산하여 변경한다. 그 다음 헤더의 실행위치 정보에 따라 세그먼트 레지스터와 IP레지스터를 지정하게 된다.

헤더의 재배치 정보에 따라 각각의 세그먼트 레지스터들이 어떻게 지정되는지? 로드 모듈내의 세그먼트 정보가 어떻게 변경되는지? 는 독자들의 혼란만 야기하는 것이 될 것 같아 여기서는 빼기로

〔그림 8〕 EXE형 화일의 구조



하고, 프로그램 작성시 주의해야 할 사항에 대해서만 언급하기로 하자.

EXE형 화일작성시 크기가 64K바이트를 넘고, 스택 세그먼트를 지정하지 않았을 경우, 프로그램 실행코드가 파괴되므로 가급적 스택은 프로그램 작성시 같이 기술해 주는 것이 좋다.

프로세스의 기동과 종료

(1) 프로세스의 기동

위에서 MS-DOS에서는 여러개의 프로세스를 수행할 수 있다고 했는데, 그렇게 여러개의 프로세스를 두었을 때, 각 프로세스의 관계와 DOS와의 관계에 대해서 알아보자. 먼저 프로세스의 기동부터 살펴보자.

●EXEC 펑션

MS-DOS에서는 COM형 화일이건 EXE형 화일이건 관계없이 하나의 실행화일을 하나의 프로세스로 생각한다. MS-DOS에서 최초로 기동하는 프로세스는 "COMMAND.COM"이고, 이 "COMMAND. COM"은 코맨드 라인을 해석하여 새로운 프로세스를 기동시킬 수 있다. 또한 모든 프로세스는 다시 또 다른 프로세스를 메모리가 허용하는 범위에서 기동시킬 수 있다.

MS-DOS는 프로세스를 기동하는 펑션으로 EXEC 펑션을 가지고 있는데, 어떤 프로세스가 EXEC 펑션을 사용해서 자프로세스를 실행할 때, 친프로세스는 디폴트화일 콘트롤 블록과 환경블록을 통해서 문자열의 정보를 보낼 수 있다.

또한, 친프로세스에서 사용되던 화일과 장치들은 새로이 생성되는 자프로세서의 작업으로 전달되며, 자프로세스에 의한 화일작업은 모프로세서에 영향을 줄 수 있다.

우리가 가끔 작업 도중에 DOS 명령을 사용해야 할 필요를 느껴 "OS Shell"을 사용하는 경우가 있는데, 이것이 바로 자프로세스를 생성하는 중요한 예가 된다.

OS Shell기능은 두번째로 "COMMAND.COM"을 메모리에 적재하여 수행하는 것을 의미하는데, 다시 말해 바로 이때 EXEC를 사용한 자프로세스로 "COMMAND.COM"을 실행하는 것이다. 우리는 이것을 이용하여 하던 작업을 종료시키지 않고서도 디렉토리를 보거나 혹은 다른 DOS 작업을 수행할 수 있다.

EXEC 펑션을 사용하여 자프로세스를 수행하기 위해서는 먼저 TPA(Transient Program Area : 임시 프로그램 영역)에 사용되지 않는 메모리가 충분해야 한다.

그런데, MS-DOS에서는 생성할 자프로세스가 요구하는 메모리양을 예측하지 못하기 때문에 친프로세스는 자신에게 할당된 메모리중 불필요한 메모리를 DOS 펑션 4AH를 사용하여 해제한다. 그런 다음 EXEC에서 필요한 정보를 준비한다(그림 9 참조).

●EXEC 펑션을 위한 정보의 생성

EXEC 펑션에 넘기는 정보로, 첫째는 기동해야 할 프로그램이 들어 있는 화일 패스명의 선두 어드레스를 DS:DX에 저장하고, 둘째로 다른 정보들의 주소를 갖고 있는 매개변수 블록의 선두 어드레스를 ES:BX에 저장한다.

패스명은 ASCII 문자열(문자열의 끝에 0코드가 더해진 형태)로 표시된다.

〔표 2〕 헤더의 구성

옵세트	내용
00H~01H	EXE 화일의 식별자(4DH, 5AH)
02H~03H	최종 페이지내의 바이트 수(여기서 페이지는 512바이트를 단위로 한다.)
04H~05H	페이지(512바이트) 단위의 헤더를 포함하는 화일, 즉 EXE 화일 자체의 크기
06H~07H	리로케이션 테이블의 항목 수
08H~09H	헤더의 크기(패러그래프 사이즈)
0AH~0BH	로드된 프로그램의 뒤에 필요한 최소 메모리영역의 패러그래프 사이즈
0CH~0DH	로드된 프로그램의 뒤에 필요한 최대 메모리영역의 패러그래프 사이즈. 이 값이 0000H의 경우 로드 모듈은 메모리의 최상위에 두어진다.
0EH~11H	프로그램 실행시의 스택값, SS : SP의 초기값
12H~13H	EXE 화일의 워드 체크섬(MS-DOS Ver. 2.0에서는 실행시에 EXE 화일의 체크섬의 검사를 하지 않음), 체크섬은 EXE 화일내의 모든 워드의 오버플로우를 무시하여 16비트로 가산했을 때 0000H가 되는 값
14H~17H	프로그램의 개시 어드레스. 14H~15H가 옵세트, 16H~17H가 세그먼트 어드레스로, 로드·모듈의 선두를 0000 : 0000H로 했을 때의 상대값으로 세트된다.
18H~19H	헤더내의 리로케이션 테이블이 시작되는 위치가 화일의 선두에 대한 옵세트로 들어간다.
1AH~1BH	오버레이번호(프로그램의 상주부는 0)가 들어간다.

확장자가 반드시 EXE나 COM이어야 하고, 와일드 카드('*', '?')를 사용할 수 없다.

매개변수 블록에는 4개의 데이터 지정 어드레스(포인터라고도 한다)가 들어 있다. 각 포인터는 환경블록, 명령어꼬리, 2개의 디폴트 FCB(File Control Block)를 지정한다.

환경블록은 EXEC 펑션에 의해서 새로운 영역에 복사되고, 복사된 환경블록의 세그먼트 어드레스는 PSP의 2CH와 2AH에 적어 넣는다. 만약에 환경블록의 포인터부를 0000H로 설정하면 EXEC 펑션은 모프로세스의 환경을 복사하여 그 환경이 그대로 계승된다(환경의 처리부분은 뒤에서 자세히 다루기로 한다).

명령어꼬리(COMMAND TAIL)는 자프로세스 PSP의 80H 부분으로 복사되는데, 첫번째 바이트는 리턴코드 0DH를 제외한 문자열의 길이가 있어야 하고, 그 다음에 실제 ASCII문자열이 오고, 마지막에는 리턴코드가 와야 한다. 전체 파라미터의 길이는 최저 2바이트(00H, 0DH)에서 최대 128바이트까지이다.

마지막으로 디폴트 FCB용 정보가 필요한데, 이 내용은 자프로세스 PSP의 5CH와 6CH에 복사될 내용이다. 만약, 디폴트 FCB를 위한 정보가 필요없다면 매개변수 블록의 FCB 포인터에 0FFFFH를 써서 생략할 수 있다.

●전처리와 후처리

여기서 한가지 주의해야 할 사항은 EXEC 펑션이 다른 펑션과 달리 CS와 IP를 제외한 모든 레지스터(SS와 SP를 포함한)의 내용을 파괴한다는 것이다. 그러므로 모든 레지스터의 내용을 스택에 저장한 후 SS와 SP레지스터를 코드 세그먼트 내부의 변수영역에 복사하고 EXEC 펑션을 수행해야 한다.

EXEC 펑션이 끝나(자프로세스의 실행이 끝남을 가리킴)고 리턴되면 SS와 SP를 복구하고 스택에 저장했던 다른 레지스터의 값을 재적재한다.

참고로 모프로세스는 자프로세스의 상태를 시스템콜의 4DH 펑션을 사용하여 얻을 수 있다.

리스트 3은 EXEC 펑션의 간단한 사용 예이다. EXEC 펑션에 넘길 정보들(패스명, 매개변수 블록 등)의 생성방법과 EXEC 펑션 사용전·후의 스택 관련 레지스터의 처리방법, 즉 SS와 SP지정 영역을 코드 세그먼트에 두어야 함에 주의해서 볼 필요가 있다.

끝으로 프로세스의 기동절차를 독자의 이해를 돕기 위해 일반적인 부분에 대해서 정리해보자.

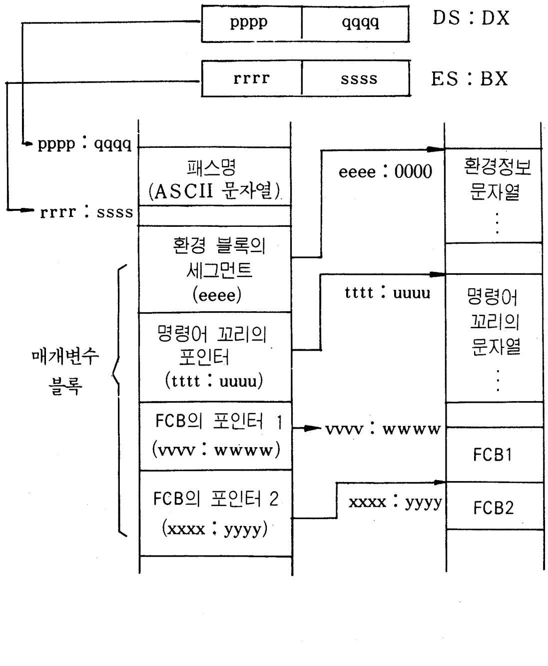
① 메모리 블록의 확보
② 모프로세스로부터 받은 환경을 자프로세스용 메모리에 복사한다.
③ PSP상에 ②항의 주소로 세트한다.
④ INT 22H의 벡터값을 종료주소(자프로세스가 종료시에 돌아갈 모프로세스의 주소)를 세트한다.
⑤ INT 22H~INT 24H의 벡터값을 PSP에 복사한다.
⑥ 메모리 용량에 관련되는 영역의 값을 세트한다.
⑦ 모프로세스로부터 받은 디폴트 FCB를 5CH, 6CH에 복사한다.
⑧ 80H부터 명령어꼬리를 복사한다.
⑨ 그밖에 필요한 영역의 값을 세트한다.
⑩ 프로그램을 적재하는데 만약 EXE형 화일이면 재배치한다.
⑪ 각 레지스터(SS : SP, DS, ES)들을 세트하고 IP를 시작번지로 한다.

(2) 프로세스 실행환경의 연속

MS-DOS상에서 작동되는 프로세스들은 각자의 실행환경을 갖고 있으며 이 실행환경들은 대부분 모프로세스 뿐만 아니라 자프로세스와도 관계를 맺고 있다.

본 절에서는 이러한 프로세스 실행환경들의 상호관계를 알아보자. 이 실행환경에는 PSP중의 코맨드 라인, CTRL-C나 치명적 에러 처리루틴의 정보, 환경, 화일핸들 등이 포함된다. 실행환경에 대한 예는 다수의 프로세스를 가질 수 있는 복잡한 응용 프로그램의 개발에서 정확한 환경의 설정과 사용 및 프로세스간의 정보교환에 관한 방법을 제공해 준다.

〔그림 9〕 EXEC 펑션의 호출규칙



특집

●프로세스와 CTRL-C, 치명적 에러 처리루틴

MS-DOS는 〈CTRL-C〉와 치명적 에러 처리루틴을 프로세스별로 설정할 수 있도록 설계되어 있다.

EXEC 펑션에 의한 프로세스 기동시 INT 23H (CTRL-C 처리루틴의 벡터), INT 24H(치명적 에러 처리루틴의 벡터)의 벡터들이 PSP에 복사되고 종료시에는 프로세스기동 이전상태가 되므로 각 프로세스에서는 독자적인 CTRL-C 처리루틴이나 치명적 에러루틴을 가질 수 있다.

그림 10의 프로세스와 INT 23H, 24H의 처리루틴에서 프로세스 2와 같이 자신의 처리루틴을 설정하지 않은 경우 모프로세스 즉, 프로세스 1의 실행환경이 이어져서 프로세스 1의 처리루틴을 공유하게 된다.

●CTRL-C에 의한 INT 23H 처리루틴

MS-DOS에서는 시스템콜(INT 21H)의 펑션 처리중에, 사용자에 의한 〈CTRL-C〉키 입력을 검사하여, INT 23H 처리루틴을 호출한다. 일반적으로 이 루틴은 실행중인 프로세스를 종료하고 모프로세스로 제어를 되돌리는 기능을 하지만 "COMMAND. COM"과 같은 프로그램이나 특수한 응용프로그램의 경우에는 CTRL-C를 무시하거나, 특별한 처리를 해야 할 필요가 있다.

INT 23H 처리루틴의 호출은 시스템콜의 처리전에 행해지므로, 그 처리루틴중에 시스템콜을 사용해도 된다. 다만 이런 경우에는 다시 CTRL-C에 의한 처리가 수행될 가능성이 있다.

리스트 4는 INT 23H 처리루틴의 작성 예로서, 이 프로그램에서는 처리루틴에서 바로 리턴하므로 〈CTRL-C〉키를 무시하게 된다.

●치명적 에러에 의한 INT 24H 처리루틴

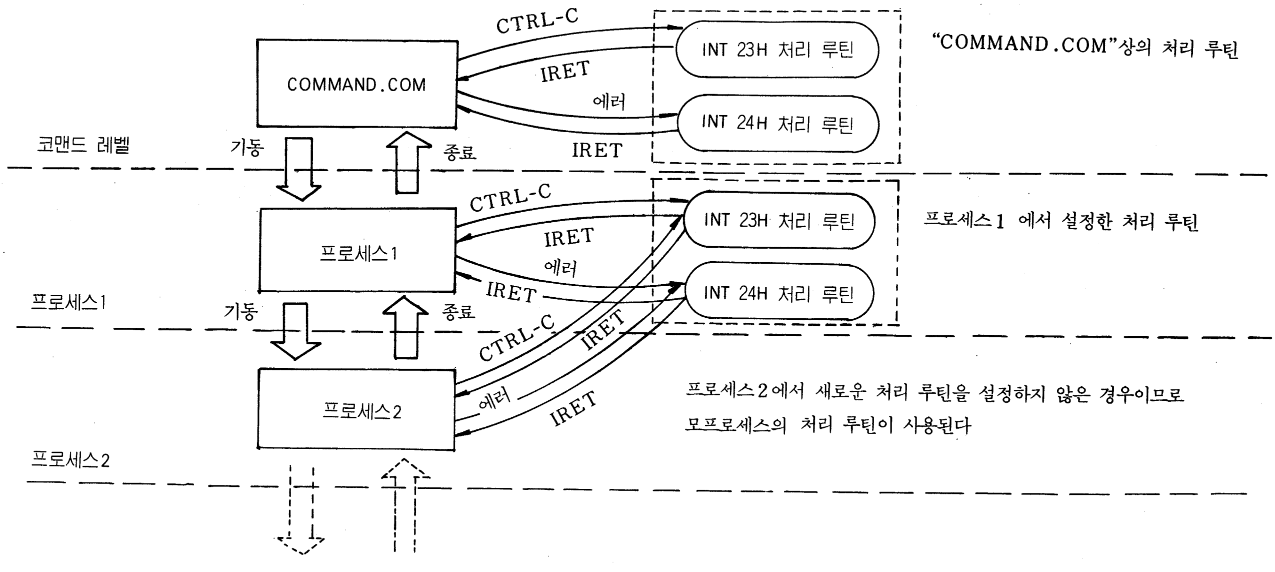
이 루틴은 MS-DOS의 시스템콜 처리중에 치명적인 에러가 발생한 경우에 발생하고, 처리가 끝나면 IRET 명령으로 돌아가야 한다. 시스템콜의 수행중에 호출되어 사용되는 것이므로 처리루틴 내부에서의 펑션 사용에 제한이 있다. 즉 펑션 01H~0CH까지는 사용이 가능하지만, 그 밖의 펑션은 DOS의 스택을 파괴시킨다. 그리고 한가지 꼭 주의해야 할 사항은 치명적 에러처리 루틴내에서 펑션을 사용할 경우, 다시 〈CTRL-C〉 처리루틴을 수행할 가능성이 있으므로 결국 사용자가 〈CTRL-C〉 처리루틴을 독자적으로 작성할 때 사용할 수 있는 펑션은 01H~0CH로 제한되게 된다.

표 3은 INT 24H 처리루틴이 호출되는 경우의 에러번호와 그 내용이다. 보통 이 루틴에서 처리하는 내용은 에러번호에 다른 에러내용을 출력하고 이후의 동작을 "Abort (A) Retry (R) Ignore(I)" 중에서 입력받아 정보를 0, 1, 2중 하나의 값으로 AL레지스터에 저장한 후 IRET명령을 사용하여 리턴한다.

특별한 처리방식으로 INT 24H 루틴에서 MS-DOS 리턴을 하지 않고 곧바로 사용자 프로그램으로 돌아갈 수 있다.

그림 11은 INT 24H루틴이 호출되었을 때의 스택상태를 보여준다. 이런 상태는 처리루틴에서 스택상의 리턴 어드레스와 플래그를 제거하고(SP레지스터의 값을 6증가하면 된다.), POP명령을 사용하여 레지스터들을 복구한 후(필요없다면 각 레지스터의 상태들도 제거해버릴 수 있다.), IRET 명

〔그림 10〕 프로세스와 INT 23H, 24H 처리 루틴



〔표 3〕 치명적 에러의 내용

번호	에 러 내 용
0	라이트 프로텍트 되어 있는 디스크에 써넣기
1	존재하지 않는 유니트 번호
2	드라이브가 준비되어 있지 않다.
3	커맨드 번호가 틀려 있다.
4	데이터 에러(CRC 에러 등)
5	코맨드 패킷의 길이가 틀려 있다.
6	검색 에러
7	미디어 타이프가 틀려 있다.
8	섹터가 발견되지 않는다.
9	프린터 용지가 없다.
10	써넣기 에러
11	읽어넣기 에러
12	그 밖의 에러

령을 사용하여 사용자 프로그램으로 되돌아 갈 수 있다.

● 환경의 계승

환경은 ASCII 문자열("ABC"는 ASCII 문자열로는 "41H, 42H, 43H, 00H"로 표시된다.)의 모양을 가지고 있는데, 하나의 ASCII 문자열은 환경변수와 그 값으로 표시되고 이것들이 모여서 환경을 형성한다. 환경의 마지막은 추가의 00H로 표시한다. 환경의 크기는 최대 32K바이트까지 가능하지만 크기에 대한 정보는 따로 존재하지 않는다. 환경은 메모리중에서 세그먼트 경계에 위치하고 그 세그먼트 어드레스는 PSP의 2CH에 저장된다(리스트 5 참조).

새로운 프로세스 기동시에 각 프로세스별로 환경을 제공할 수 있지만, 일반적으로는 모프로세스의 환경을 복사하여 계승받는다. 여기서 중요한 점은 각각의 프로세스별로 환경의 복사본을 갖고 있으므로, 프로그램 수행중에 환경이 변경되어도 모프로세스에 영향을 미치지 못한다는 점이다.

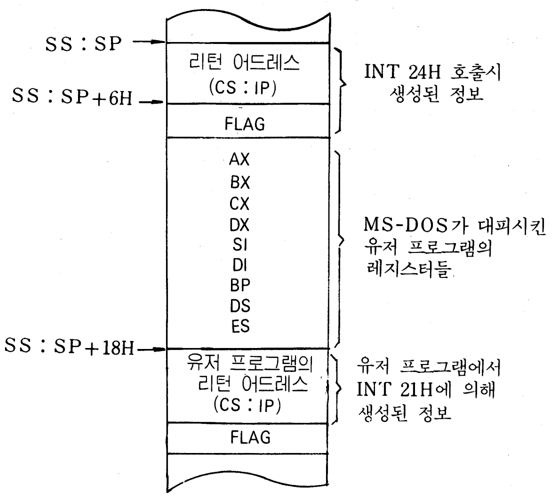
환경을 변경할 때, 주의할 점이 있는데 이는 아래와 같다. 보통, 환경은 PSP의 앞부분에 존재(프로세스 기동시 환경 복사후 그 뒤에 PSP를 형성)하므로 축소할 때는 별 문제가 없지만 환경을 확장해야 할 경우 PSP가 파괴되지 않도록 하기 위하여 프로세스가 관리하는 영역에 복사한 후 그 뒤에 환경을 추가하고 PSP 2CH의 값을 변경해야 한다(그림 12 참조).

자프로세스에 새로운 환경을 설정하는 방법은 "프로세스 기동"편에서 설명했으니 참조하기 바란다.

● 화일핸들의 계승

프로세스간에 이어지는 실행환경으로서 PSP나

〔그림 11〕 INT 24H 발생시 스택의 상태



환경 이외의 화일핸들이 있다. 화일핸들은 MS-DOS에서 화일을 액세스하는 2가지 방식, FCB와 화일핸들 중의 하나로 마이크로소프트사에 의해서 권장되고 있는 화일 관리방식을 들 수 있다.

본고에서는 FCB의 특징과 차이점에 관해서만 간단히 언급하고, 실행환경이란 측면에서 화일핸들을 설명하겠다. FCB 개념은 CP/M에서 온 것으로, 이는 MS-DOS 개발 역사와 관계가 있다(본지 3월호 특집 참조).

FCB는 MS-DOS 버전2.0 이전에 채용되었으나, 그 이후에는 유닉스의 구조를 도입한 화일핸들 개념을 추가하였다. FCB의 사용은 2가지 목적이 있는데, 첫째는 화일의 오픈, 클로즈, 읽기, 쓰기를 위해 사용될 수 있고, 둘째로는 화일의 검색과 삭제 등을 행하는데 그 목적이 있다. 후자의 경우에는 복수의 화일을 대상으로 하는 와일드카드('?','*')를 사용할 수 있다는 장점이 있어 현재에도 자주 사용하고 있으나 전자의 경우에는 단점이 많아 현재에는 사용되고 있지 않는 실정이다.

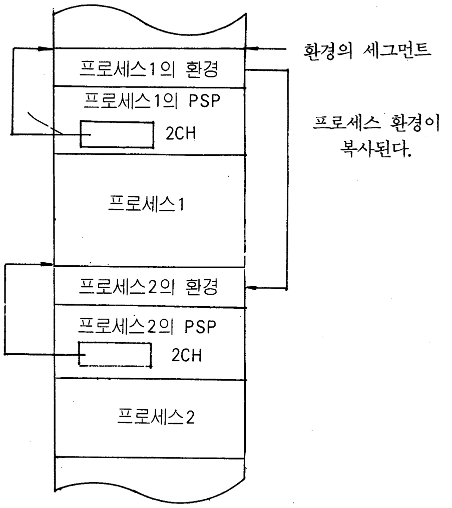
FCB를 사용해 화일을 액세스할 경우 상당히 번거로운 작업이 필요하다. 즉, FCB에 화일명을 세트하고 레코드 사이즈와 커런트 블록, 커런트 사이즈를 바르게 세트해야 하며 랜덤 액세스의 경우에는 상대 레코드를 써야 하는 불편이 뒤따르게 된다.

결국, 화일관리를 위해서 블록과 레코드 관리까지 해야하는 엄청난 불편을 감수해야 한다. 또한 FCB는 계층적 디렉토리 구조를 지원하지 못하므로 버전2.0 이후에는(이때부터 계층적 디렉토리 구조가 도입되었다.) 화일핸들 사용이 권장되고 있다.

그럼, 이제부터는 화일핸들에 관해서 알아보자.

화일핸들이란? 2바이트의 정수로써 화일을 오픈하거나 작성할 때, "MSDOS.SYS" 측에서 넘겨주

〔그림 12〕 환경의 복사와 배치



는 화일번호이다.

화일의 오픈은 ASCII 문자열로 만들어진 화일명(디렉토리 정보포함 가능)의 포인트를 갖고, 시스템콜 펑션 3DH를 호출하면 되고, "MSDOS.SYS"는 해당 화일을 찾아 화일핸들을 돌려준다. 그 이후에 사용자 프로그램은 2바이트 화일핸들만을 관리하면 된다. 한편 어떤 화일에 대한 화일핸들, 즉 화일번호의 할당은 "MSDOS.SYS"에서 관리한다. 기본적으로 "MSDOS.SYS"는 0에서 4까지의 화일핸들을 오픈해 놓는데, 이를 일컬어 표준 화일핸들이라 한다. 그 내용은 **표 4** 표준 화일핸들에서 보여준다.

실행환경으로서 화일핸들의 관계는 아주 단순하게 설계되어 있다.

모프로세스에서 오픈한 화일핸들은 그대로 자프로세스에 이어진다. 그러므로 화일명에 관계없이, 화일핸들만으로 입출력이 가능해진다(**리스트 6** 참조). 이러한 화일 계층구조를 가장 효과적으로 사용한 것이 재지정 개념이다.

"COMMAND.COM"에서는 표준입력(화일핸들 0)과 표준출력(화일핸들 1)에 재지정에 의한 화일을 할당하고, 자프로세스를 기동하면 화일핸들이 이어지므로 자프로세스에서는 표준입력이 콘솔이건 화일이건 관계없이 동일한 동작으로 재지정처리가 이루어진다.

또한 재지정의 기능은 시스템콜 펑션 45H와 46H를 사용하여 그 기능을 제공받을 수 있다.

(3) 프로세스의 종료

프로세스의 종료는 아래와 같은 처리과정을 밟아 프로세스 기동전의 상태로 되돌아간다.

먼저, 현재 오픈된 화일을 모두 클로즈하고, PSP의 오프셋 0AH로부터 3개의 더블워드 포인터를 INT 22H~24H의 벡터영역으로 되돌린다. 그리고 환경이나 PSP 프로그램 자체가 점유하고 있던 메모리를 차지하고, INT 22H의 벡터 또는 PSP의 오프셋 0AH에 있던 벡터로 제어권을 넘긴다.

프로세스 종료의 방법으로는 아래와 같이 상주종료와 비상주 종료로 나누어지는데 이를 살펴보면 아래와 같다.

●상주 종료

상주 종료는 볼랜드사의 사이드킥이나 삼보컴퓨터의 NKP나 THP와 같은 많은 유틸리티 또는 어플리케이션 프로그램에서 주로 사용하고 있는데, 그것들은 종료시 점유하고 있던 메모리를 개방하지 않고, 아래의 2가지 방법 중 하나를 이용하여 메모리에 상주시키고 있다.

① AH=31H로 하여 DX레지스터에 상주하는 패러그래프의 사이즈를 넣어 INT 21H를 실행한다.
② DX레지스터에 상주시킬 바이트수를 넣어 INT 27H를 실시한다.

단, 두번째 방법의 INT 27H를 사용할 경우 64K

〔표 4〕 표준 화일 핸들

번호	핸 들	기 능
0	표준 입력	보통 CON으로 되어 있는데, 재지정에 의해 화일로 할 수 있다. 프로그램에 대한 디폴트의 입력을 정한다.
1	표준 출력	보통 CON인데, 화일로 교체할 수도 있다. 프로그램이 출력하는 문자(열)를 어디로 출력할지를 정한다.
2	표준 에러 출력	항상 CON으로 되어 있고, "COMMAND.COM"에서는 재지정되지 않는다. 이 때문에 표준 출력이 화일로 되어 있어도 이 표준 에러 출력에 문자열을 출력하면, 그 문자열이 스크린에 출력된다. 에러 메시지의 출력에 적합하다.
3	표준 보조장치	RS-232와 같은 디바이스를 가리킨다. 시스템에 따라서 이 디바이스는 옵션으로 되어 있다. 디바이스명은 AUX. 재지정은 기계어 레벨에서 가능.
4	표준 프린터	프린터에 문자열의 출력용 할당번호. 디바이스명 PRN에 상당하다. 기계어를 사용하면 재지정이 가능

바이트 이상을 상주시킬 수는 없다.

●비상주 종료

비상주 종료의 방법으로는 아래와 같은 것들이 있다.

① PSP의 오프셋 00H로 점프한다(**리스트 7** 참조).
② CS레지스터에 PSP의 선두 어드레스를 세트하고 INT 20H를 실시한다.
③ AH=00H로 하여 CS레지스터에 PSP의 선두 어드레스를 세트하여 INT 21H를 실시한다.
④ AH=4CH로 하여 INT 21H를 실시한다.
⑤ AH=00H 또는 4CH로 하여 PSP의 오프셋 50H로 콜한다.

시스템콜의 종류와 사용방법

MS-DOS의 시스템 서비스의 대부분은 내부인터럽트 21H에 의해서 제공되어 지는데 그 기능은 디렉토리에 관한 것, 화일에 관한 것 등 여러가지를 제공하고 있다. 또한 이 INT 21H는 표준화 되어 있어 어떤 MS-DOS 시스템에서도 이를 이용하여 제작한 프로그램은 실행이 가능하다.

(1) 시스템콜의 사용방법

MS-DOS 시스템콜을 하는 방법에는 여러가지 있다. 그중 가장 일반적으로 사용하는 것은 AH레지스터에 기능번호를 다른 레지스터에는 그 기능의 파라미터를 부여하고 INT 21H 명령을 사용하는 방법이 있다. 또 잘 쓰이지는 않지만 PSP의 오프셋 50H를 콜하는 방법도 있고 PSP의 오프셋 05H를 콜하는 방법도 있다.

(2) 시스템콜의 종류

① 콘솔과의 입출력 AH=02H, 06H, 09H, 0AH-0BH
② 디바이스와의 입출력 AH=03H-05H
③ FCB에 의한 화일 액세스 AH=0DH-24H, 27H-29H, 2FH
④ 시간/플래그에 관한 제어 AH=25H, 2AH-2EH, 32H-38H, 54H
⑤ 화일핸들에 의한 화일 액세스 AH=3CH-46H, 4EH, 56H, 57H
⑥ 디렉토리 관리 AH=39H-3BH, 47H
⑦ 프로세스 관리 AH=00H, 31H, 4BH-4DH
⑧ 메모리 관리 AH=48H-4AH

프로그램 작성에 대한 지침

본장의 내용은 마이크로소프트사에서 MS-DOS의 버전 업에 대한 호환성의 유지를 위하여 상기시켜주는 지침서의 내용중에서 본고의 내용과 연관성있는 부분을 발췌하여 구성하였다.

(1) 인터럽트 처리

- INT 22H(프로세스의 종료)를 외부적(응용 프로그램에서 호출하는 것)으로 사용하지 마라. DOS만이 이것을 사용해야 한다. 사용자가 종료번지를 변경해야 할 경우에는 펑션 35H(인터럽트 벡터를 얻음)과 펑션 25H(인터럽트 벡터의 설정)을 사용하여 INT 22H의 벡터 케이블의 내용을 변경하도록 한다.
- INT 24H의 처리루틴은 ES레지스터를 보존해야 하며, 특히 재개(Retry)나 무시(Ignore)할 경우 SS, SP, DS, BX, CX, DX를 보존해야 한다.
- INT 23H는 응용 프로그램에서 호출하지 마라.

(2) 시스템 펑션의 호출

- 새로운 시스템 호출을 사용하라. 주로 화일 액세스에 관한 내용으로 FCB의 사용을 지양하고, 화일핸들의 사용을 권장한다. 프로세스 종료시에도 INT 20H나 펑션 00H보다 펑션 4CH를 사용하는 것이 유리하며, 새로운 프로세스 생성을 위하여 펑션 26H(새로운 PSP 작성)대신 펑션 4CH(EXEC 펑션)를 사용하는 것이 좋다.

(3) 메모리 관리

- 할당된 메모리만 사용하라. MS-DOS에서는 메모리 관리를 위해 3개의 펑션(48H : 메모리할당, 49H : 할당된 메모리해제, 4AH : 블록설정)을 사용한다. 할당되지 않은 메모리를 사용하는 경우, 다른 프로세스의 메모리 제어블록을 파괴하거나 다른 프로세스에 의해서 파괴될 수 있다.

(4) 프로세스 관리

- 프로세스의 기동은 EXEC 펑션을 사용하지 마라. EXEC 펑션은 중복 로드(Overlay)처리가 가능하고, EXE 화일형태의 변화 등 버전에 의한 변화에서 독립될 수 있다.
- INT 27H(프로그램의 상주종료) 대신 펑션 31H를 사용하라. 펑션 31H는 64K바이트 이상의 프로그램을 상주시킬 수 있다.
- 프로세스의 종료는 펑션 4CH를 사용하라. 이 펑션은 상위 프로세스인 현 프로세스의 상태를 리턴코드를 통해서 전달할 수 있게 하므로 프로세스의 정보교환에 유용하다. PC

〔리스트 1〕 PSP의 예

```
C:\>debug command.com ↵
-d 0 100 ↵
26BA:0000  CD 20 00 A0 00 9A F0 FE-1D F0 F4 02 EE 22 2F 03   . ..........."/.
26BA:0010  EE 22 BC 02 EE 22 B6 22-01 01 01 00 02 FF FF FF   ."..."."........
26BA:0020  FF FF FF FF FF FF FF FF-FF FF FF FF AE 26 46 01   .............&F.
26BA:0030  28 26 14 00 18 00 BA 26-FF FF FF FF 00 00 00 00   (&.....&........
26BA:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
26BA:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20   .!...........
26BA:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20        .....
26BA:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00        ........
26BA:0080  00 0D 63 6F 6D 6D 61 6E-64 2E 63 6F 6D 0D 00 0D   ..command.com...
26BA:0090  3B 63 3A 5C 75 74 69 6C-5C 6E 6F 72 74 6F 6E 3B   ;c:\util\norton;
26BA:00A0  63 3A 5C 75 74 69 6C 5C-6E 6F 72 74 6F 6E 5C 6E   c:\util\norton\n
26BA:00B0  63 3B 63 3A 5C 75 74 69-6C 5C 70 63 74 6F 6F 6C   c;c:\util\pctool
26BA:00C0  73 3B 63 3A 5C 75 74 69-6C 5C 73 68 61 72 65 3B   s;c:\util\share;
26BA:00D0  63 3A 5C 75 74 69 6C 5C-70 63 6D 61 67 61 3B 63   c:\util\pcmaga;c
26BA:00E0  3A 5C 75 74 69 6C 5C 76-61 63 63 69 6E 65 0D 00   :\util\vaccine..
26BA:00F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
26BA:0100  E9
```

〔리스트 2〕 헤더의 예

```
C:\>ren exe2bin.exe exe2bin.com ↵
헤더를 보기위해 화일 이름을 바꾼다.
C:\>debug exe2bin.com ↵
-d 100 ↵  100H 전 까지는 PSP이므로 100H이후에 헤더가 존재한다.
268D:0100  1E 33 C0 50 B4 30 CD 21-86 E0 3D 14 03 74 11 BB   .3.P.0.!..=..t..
268D:0110  D6 26 8E DB BA 5C 01 B4-09 CD 21 06 33 C0 50 CB   .&...\....!.3.P.
268D:0120  BE 81 00 BB D6 26 8E C3-8B 1E 02 00 E8 2B 01 73   .....&.......+.s
268D:0130  08 06 1F BA 91 01 E9 6A-01 BF 19 00 AC E8 2D 01   .......j......-.
268D:0140  74 28 26 8B 16 16 00 3C-5C 74 04 3C 3A 75 0C 26   t(&....<\t.<:u.&
268D:0150  89 3E 66 00 26 FF 06 66-00 33 D2 3C 2E 75 03 8B   .>f.&..f.3.<.u..
268D:0160  D7 4A 26 89 16 16 00 AA-EB D2 4E 26 C6 05 00 E8   .J&.......N&....
268D:0170  E8 00 73 1D 26 80 3E 1A-00 3A 75 12 26 A1 19 00   ..s.&.>..:u.&...
-q↵
C:\>ren exe2bin.com exe2bin.exe↵
```

〔리스트 4〕 INT 23H 처리 루틴의 작성 예

```
         mov   ah, 25h              ; 시스템콜 펑션 25H를 사용하여,
         mov   al, 23h              ; 인터럽트 23H의 값을 새로 정의한
         mov   dx, seg CTRL-C       ; CTRL-C 처리루틴으로 바꾸어 준다.
         mov   ds, dx               ;
         mov   dx, offset CTRL-C    ;
         int   21h                  ;
         jmp   skip
CTRL-C:  iret                       ; 새로 정의한 <CTRL-C>처리 루틴은
 skip:                              ; 아무 처리도 하지 않는다.
```

〔리스트 3〕 EXEC 사용의 예

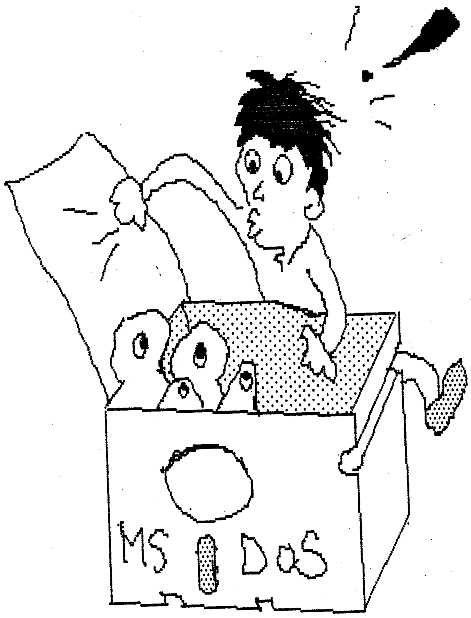
```
Vector    struc

   _offset  dw    ?
   _segment dw    ?

Vector    ends
        .
        .
        .
    mov  cs:StackSegment, ss
    mov  cs:StaskPointer, sp
    mov  dx, offset PgmName
    mov  bx, offset ParaBlock
    mov  al, 0
    mov  ah, 4Bh
    mov  ah, 4Bh
    int  21h
    mov  ss, cs:StackSegment
    mov  sp, cs:StackPointer
        .
        .
        .

StackSegment   dw     0
StackPointer   dw     0

ParaBlock      Vector      <,envir>
CommandLine    Vector      <offset CmdLine, seg CmdLine>
FileControlB1  Vector      <offset FCB1, seg FCB1>
FileControlB2  Vector      <offset FCB2, seg FCB2>
```



〔리스트 5〕 환경의 예

```
C:\DOS>DEBUG↵
-d 0↵
2744:0000  CD 20 00 A0 00 9A EE FE-1D F0 F4 02 95 23 2F 03   . ...........#/.
2744:0010  95 23 BC 02 95 23 43 0C-01 01 01 00 02 FF FF FF   .#...#C.........
2744:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 55 23 4E 01   ............U#N.
2744:0030  CF 26 14 00 18 00 44 27-FF FF FF FF 00 00 00 00   .&....D'........
2744:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2744:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 20 20 20      .!...........
2744:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20           .....
2744:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00           ........
-d 2355:0↵
2355:0000  43 4F 4D 53 50 45 43 3D-43 3A 5C 43 4F 4D 4D 41   COMSPEC=C:\COMMA
2355:0010  4E 44 2E 43 4F 4D 00 50-52 4F 4D 50 54 3D 24 70   ND.COM.PROMPT=$p
2355:0020  24 67 00 50 41 54 48 3D-43 3A 5C 55 54 49 4C 49   $g.PATH=C:\UTILI
2355:0030  54 59 5C 50 43 54 4F 4F-4C 53 3B 43 3A 5C 55 54   TY\PCTOOLS;C:\UT
2355:0040  49 4C 49 54 59 5C 4D 41-47 41 5A 49 4E 45 00 00   ILITY\MAGAZINE..
2355:0050  01 00 43 3A 5C 44 4F 53-5C 44 45 42 55 47 2E 43   ..C:\DOS\DEBUG.C
2355:0060  4F 4D 00 00 00 00 00 00-00 00 00 00 00 00 00 00   OM..............
2355:0070  5A 5D 23 A3 7C 00 00 00-00 00 00 00 00 00 00 00   Z]#.¦...........
-q↵
C:\DOS>
```

〔리스트 6〕 화일핸들 계승의 예제 프로그램

```
set_block   macro  last_byte
                                          ;COM 프로그램의 초기 메모리
        mov     bx, offset last_byte     ;할당 블럭을 감소시킨다.
        mov     cl, 4                    ;
        shr     bx, cl                   ;
        add     bx, 17                   ;
        mov     ah, 4ah                  ;4AH 펑션은 메모리 할당
        int     21h                      ;블록의 크기를 변경시킨다.
        mov     ax, bx                   ;ES는 메모리 블록의 세그먼트
        shl     ax, cl                   ;어드레스이고, BX는 메모리
        dec     ax                       ;블록의 새로운 크기(패러
        dec     ax                       ;그래프 단위)이다.
        mov     sp, ax                   ;
        endm


create_handle  macro path, attrib        ;새로 작성할 화일의 디렉
                                          ;토리 패스와 그 화일속성을
        mov     dx, offset path          ;각각 DX와 CX에 건네준다.
        mov     cx, attrib               ;DX는 ASCII 문자열의 오프
        mov     ah, 3ch                  ;셋이다. 펑션 사용후
        int     21h                      ;리턴되는 것은 새로 할당
                                          ;된 화일핸들이다.
        endm                             ;


xdup   macro   handle                    ;
                                          ;BX에 저장된 오픈된 화일
       mov     bx, handle                ;핸들을 하나 더 만든다.
       mov     ah, 45h                   ;이 펑션 사용후 AX에는
           int     21h                   ;새로 작성된 핸들이
                                          ;저장된다.
           endm                          ;


xdup2  macro handle1, handle2            ;
                                          ;이 기능은 BX에 지정된
           mov     bx, handle1           ;핸들과 CX에 지정된 두번
           mov     cx, handle2           ;째 핸들로서 같은 화일을
           mov     ah, 46h               ;가리키게 한다.
           int     21h                   ;종료후, CX에 있는 핸들은
                                          ;BX에 있는 핸들이 가리키
           endm                          ;는 화일을 지목한다.

exec     macro   path, command, parm     ;매크로는 프로그램을
                                          ;로드하여 실행. DX에는
         mov  dx, offset path            ;실행할 프로그램의 패스가,
         mov  bx, offset parm            ;BX에는 파라미터 블록의
         mov  word ptr parm[02h], offset command;오프셋이 존재.
         mov  word ptr parm[04h], cs     ;PSP의 02H와 04H에는 세
         mov  word ptr parm[06h], 5ch    ;PSP의 건네줄 명령어 꼬리의
         mov  word ptr parm[08h], es     ;어드레스가 존재. 06H와 08H
         mov     word ptr parm[0ah],6ch;에는 새 PSP의 5CH에 위치할
         mov     word ptr parm[0ch],es;FCB의 어드레스가 존재한다.
         mov     al, 0                   ;0AH와 0CH에는 두번째 FCB가
         mov     ah, 4bh                 ;있다.
         int     21h                     ;
                                          ;
         endm                            ;
close_handle macro handle                ;
```

```
                                ;이 기능은 3CH나 3DH에
        mov     bx, handle      ;의해 열린 화일핸들을
        mov     ah, 3eh         ;닫는다. 이때, BX에는
        int     21h             ;닫고자하는 열린 화일핸들
                                ;이 있어야 한다.
        endm                    ;

errmsgout macro msg             ;
                                ;"$"로 끝나는 스트링을
        mov     dx, offset msg  ;표준출력으로 내보낸다.
        mov     ah, 9           ;
        int     21h             ;
                                ;
        endm                    ;


code    segment
        assume  cs:code, ds:code, es:code

        org  100h

pgm_file     db  'command.com',0       ;이 프로그램은 표준출력을
cmd_line     db  9, '/c dir /w', 0dh   ;DIRFILE이라는 화일로 방향
parm_blk     db  14 dup (0)            ;을 바꾸고,디렉토리를 얻기
path         db  'dirfile',0           ;위하여 두번째 COMMAND.COM
dir_file     dw  ?                     ;을 실행한다.
sav_stdout   dw  ?
;
main:
        mov     ax, cs
        mov     ds, ax
        mov     es, ax
;
        set_block last_inst
        jnc     skip
        jmp     error
;
skip:   create_handle path, 0
        jc      error
;
        mov     dir_file, ax
        xdup    1
        jc      error
;
        mov     sav_stdout, ax
        xdup2   dir_file, 1
        jc      error
;
        exec    pgm_file, cmd_line, parm_blk
        jc      error
;
        xdup2   sav_stdout, 1
        jc      error
;
        close_handle sav_stdout
        jc      error
;
        close_handle dir_file
        jc      error
TheEnd:
        mov     ax, 4c00h
        int     21h
errmsg  db      'Error - Aborting $'
error:
        errmsgout   errmsg
        jmp     TheEnd

last_inst:
code    ends
        end  main
```

〔리스트 7〕 첫번째 방법에 의한 종료

```
StackSeg    segment  'STACK'
            db    128 dup(?)
StackSeg    ends

Code        segment
            assume cs:Code, ds:Work, ss: StackSeg

 FarRet     proc   far   ; FAR를 지정하여 FAR 리턴으로 종료할 수 있게
            push   es    ; PSP의 세그먼트 어드레스를 저장한다.
            mov    ax, 0 ;
            push   ax    ; 이렇게 처리하면 PSP의 오프셋 00H가 리턴 어드레스

            mov    ax, Work
            mov    ds, ax
            mov    dx, offset Msg
            mov    ah, 09h
            int    21h
 ;
            ret    ; FAR 리턴으로 되돌아오면 CS에 PSP의 세그먼트 어드레
 FarRet     endp   ; 대입하여 PSP의 오프셋 00H로 점프하게되어 시스템으.

Code        ends

Work        segment
   Msg      db   ' 퍼스널 컴퓨터 $'
Work        ends

            END FarRet
```

발송문 관리

임정윤 /한국팔란티어(주)

발송문 기능은 여러가지의 용도로 사용하는 것이 가능하다. 즉 회사를 이전했을때 보내는 이전 안내장, 새로운 상품이 나올 때마다 보내야 하는 신상품 안내서, 세미나, 전시회 개최에 따르는 초대장 발송, 연말 연시의 연하장 및 감사카드 보내기 등 남에게 알리고 전해야 하는 일들이 끊임없이 생겨난다.

이렇게 여러사람에게 알리고자 하는 소식이 있을때 우리는 보통 편지형식으로 작성하여 우편으로 보낸다.

이런 경우 직접 사람이 작성한다면 내용은 같지만 각기 다른 사람에게 보내져야 하는 이유로 각각의 수신인을 바꿔가며 문서를 만들어야 했다. 우리는 그동안 수신인 자리는 비운채 내용만을 여러장 복사하여 각각의 수신인을 써넣는 방법이라든가, 아니면 워드프로세서를 이용하여 인쇄할 때마다 수신인을 수정하여 인쇄하는 등의 방법으로 작업을 해왔다.

두가지 방법 모두 몹시 번거로운 작업이다. 게다가 수신인에 따라서 문서의 내용을 조금씩 바꿔야 할 경우에는 일일이 작성하여 사용하는 것과 시간적인 차이가 별로 없다. 이런 사용자들의 고민을 해결할 수 있는 기능이 팔란티어 워드프로세서 내의 발송문 기능이다.

발송문 기능은 조금만 신경써서 연습해 두면 일상업무에서 유용하게 사용할 수 있는 기능이다.

발송문 기능을 사용하려면 일반적으로 2개의 화일이 필요하다. 하나의 화일은 발송될 내용을 입력한 발송문 화일이고 다른 하나의 화일은 수신인의 데이터를 가지고 있는 자료화일이다.

이렇게 별개의 개념으로 작성한 2개의 화일은 발송문 문서 상단에 특정 명령행을 삽입함으로써 인쇄시 자료 화일에 있는 데이터를 차례로 읽어서 발송문 문서의 원하는 위치에 삽입되어 출력하게 된다. 인쇄시에는 작성된 문서를 그대로 출력할 수 있고, 보낼 대상에 따라 데이터를 선별하여 인쇄하는 것이 가능하다. 그리고 한번 작성된 자료화일은 같은 대상에게 다른 내용의 문서를 다시 발송해야 하는 경우 아무런 수정없이 그대로 사용하거나 DBASEⅢ+로 작성된 데이터 화일(.DBF)을 자료화일로 이용할 수도 있다.

이제부터 자료화일과 발송문 화일의 작성, 인쇄하는 방법, 데이터 베이스화일을 이용하는 방법 등을 알아보자.

자료화일의 작성

발송문 화일에서 사용하고자 하는 수신인에 대한 자료를 하나의 화일로 만든 것이다.

팔란티어 워드프로세서 편집모드에서 화면 1의 내용을 입력한다. 입력시 주의해야 할 점은 각행의 끝에서 반드시 RETURN 키를 눌러주어야 한다는 것이다. 그 이유는 발송문 자료 화일에서는 1행을 1개의 데이터(필드)로 취급을 하

〔화면 1〕 자료화일작성 화면

```
135-010                        ┐
강남구 논현동 365-1            │ 1st 레코드
논현물산(주)                   │
대리                           │
박 기수                        │
2                              ┘
150-010                        ┐
영등포구 여의도동 ABC아파트    │ 2nd 레코드
                               │
                               │
김 정훈                        │
1                              ┘
121-020
마포구 공덕동 170번지
세계소프트웨어
과장
이 영준
3
150-010
영등포구 여의도동 30번지
하나상사(주)

강 만수
2
```

는데 그 한행이라는 구분이 바로 RETURN키에 의해서 생기기 때문이다.

입력이 다 끝났으면 ESC키를 눌러 기본메뉴 상태로 빠져나온뒤 MAIL.DAT라는 이름으로 작성한 자료화일을 보관한다. 자료화일로 사용하기 위해 만든 문서는 보관할 때 확장자를 반드시 DAT라고 지정해야 한다.

한 사람(회사)분의 데이터를 우리는 1개의 레코드라 한다. 화면 1에서 보듯이 레코드와 레코드의 구분은 특별히 없다. 단지 각 레코드는 동일한 갯수의 행으로 그리고 동일한 순서에 입각하여 작성하여야 한다.

만약 해당 데이터가 없으면 빈행을 삽입하여 행수를 맞추어야 한다. 그 이유는 발송문 화일에서 MAIL.DAT화일을 사용할 때 일정 갯수의 행을 1개의 레코드로 구분하기 때문이다. 위의 자료 화일을 보면 1행은 우편번호, 2행은 주소, 3행은 회사이름, 4행은 직위, 5행은 이름, 6행은 어떤 과목을 배운 사람인지를 구분하기 위한 것으로 1은 워드프로세서, 2는 DBASE, 3은 LOTUS 과목의 수강자를 의미한다.

지금 예에서는 4개의 레코드만을 작성했는데 독자가 필요하다면 얼마든지 더 추가할 수 있다.

문서화일 작성

자료화일이 완성되었으면 이 자료 화일을 이용하여 발송할 문서를 만들어야 한다.

화면 2의 내용대로 입력한 뒤 MAIL1.WP로 보관한다. C

〔화면 2〕 문서화일작성 화면

```
CFILE "MAIL.DAT" L(6)                                        1)
CREAD ZIP, ADDRESS, COMPANY, POSITION, NAME, CHECK           2)

VNAME VPOSITION 님께                                         3)


CPRINT 2 IF CHECK = 1                                        4)
   대성 소프트웨어(주)에서는  워드프로세서 교육
   수강자를 대상으로 3월 10일 오후 4시 특강을 실시합니다.
CPRINT 2 IF CHECK = 2                                        5)
   대성 소프트웨어(주)에서는 DBASEIII+ 교육 수강자를 대상
   으로 3월 11일 오후 4시 실무 활용을 위한 특강이 있읍니다.
CPRINT 2 IF CHECK = 3                                        6)
   대성 소프트웨어(주)에서는 여러분의 LOTUS 실무 활용을
   돕기위해 3월 12일 오후 4시 공개 토론회를 개최합니다.
                                                         ┐ 7)
   관심있는 분들의 많은 참석을 바랍니다.                 │
                                                         ┘
```

는 (F1과 C를 누른다) 그행은 명령행이라는 표시가 되고, V변수명은 (F1과 V를 누른다) 해당 변수명의 내용을 인쇄하라는 의미로 두개는 발송문 기능에서만 사용되는 명령어이다.

화면 2에서 1)번과 2)번행은 해당 자료화일과 연결시켜 인쇄하기 위한 명령이고 그 이후는 발송할 문서의 내용이다. 1)번과 2)번행을 보면 굵게 표시된 부분과 보통 인쇄부분 2개로 구분되어 있다.

굵은 글씨로 표시된 부분은 발송문 작성시마다 변하는 부분이고 보통 글씨 부분은 항상 그대로 써줘야 하는 부분이다. 이렇게 고정적으로 써줘야하는 명령어를 우리는 예약된 명령어라고 한다. 이 두개의 명령행에 의해서 인쇄시 이 문서가 발송문 기능을 사용하는 문서라는 인식을 하게 된다.

문서 중간에 삽입되어 있는 명령행은 문서를 조건에 맞추어 인쇄하기 위한 조건 명령어이다. "V변수명"은 문서 중간에 해당 변수의 내용을 삽입하고자 하는 위치에서 해주면 된다.

그럼 각행의 기능을 자세히 알아 보기로 하자.

1) 문서가 인쇄될 때 MAIL.DAT라는 화일에서 6행씩 취하라. 굵게 표시된 글자는 그때마다 바뀌는 부분이고 나머지는 예약된 명령어이다. 인용부호 안의 화일이름은 함께 연결시켜 사용할 자료화일 이름을 지정하는 것이고 괄호안의 숫자는 1레코드에 해당하는 행수를 의미한다.
2) 첫행에 의해서 취해진 6행의 데이터를 각각 ZIP, ADDRESS, COMPANY, POSITION, NAME, CHECK라는 변수로 읽어들여라. READ는 예약된 명령어이고 그뒤에 컴머로 구분된 단어를 우리는 변수라고 한다. 변수명은 영문 8자 이내로 사용자가 임의로 정의하면 되고, 각각의 변수는 읽어들인 데이터를 임시로 기억하고 있는 기능을 수행한다. 그러므로 READ 뒤에 나열된 변수의 갯수는 위에서 정의한 괄호안의 숫자와 일치하여야 한다.
3) NAME변수에 있는 내용과 POSITION 변수에 있는 내용을 인쇄하라.
4) CHECK변수의 값이 1이면 다음 2행을 인쇄하라. "PRINT n IF 조건" 명령은 IF 뒤의 조건이 만족하면 다음 n행을 인쇄하라는 조건 명령이다.
5) CHECK변수의 값이 2이면 다음 2행을 인쇄하라.
6) CHECK변수의 값이 3이면 다음 2행을 인쇄하라.
7) 아무런 조건이 없으므로 그대로 인쇄된다.

위의 4) 5) 6)번의 명령에 의해서 WP, DBASE, LOTUS 교육자에게 보내기 위한 각기 다른 내용의 문서가 출력된다. 발송한 문서가 완성 되었으면 봉투도 발송문 기능을 이용하여 인쇄해 보기로 하자.

화면 3의 내용을 입력한 뒤 MAIL2.WP로 보관한다. MAIL-2. WP에 대한 내용은 큰 어려움없이 알 수 있을 것이다.

〔화면 3〕 자료화일의 출력 양식 지정화면

```
CFILE "MAIL.DAT" L(6)                                    1)
CREAD ZIP, ADDRESS, COMPANY, POSITION, NAME, CHECK       2)

150-010

영등포구 여의도동 36-2
대성소프트웨어(주)



                          VZIP                           3)

                          VADDRESS                       4)
CPRINT IF COMPANY                                        5)
                          VCOMPANY                       6)
                          VNAME VPOSITION 님 귀하        7)
```

1) 문서가 인쇄될 때마다 MAIL.DAT라는 자료화일에서 6행씩 취하라.
2). 자료화일에서 취한 6행의 데이터를 나열된 변수명에 차례로 읽어 들여라.
3) ZIP 변수의 내용을 인쇄하라.
4) ADDRESS 변수의 내용을 인쇄하라.
5) COMPANY 변수에 내용이 있을 때만 다음 1행을 인쇄하라. 회사에 소속되어 있지 않은 사람은 COMPANY변수에 해당하는 DATA가 없다. 이런 경우에 그 다음행이 빈행으로 인쇄되는 것을 방지하기 위한 명령이다.
6) COMPANY변수의 내용을 인쇄하라.
7) NAME변수와 POSITION 변수의 내용을 인쇄하라.

MAIL1.WP와 MAIL2.WP가 모두 완성되었으면 이제 2개의 문서를 차례로 인쇄해 보기로 하자.

낱장 자동공급장치가 부착되어 있는 프린터라면 각각의 문서를 낱장에 인쇄해 낼 수 있지만, 그렇지 않을 경우에는 연속용지에 인쇄를 해서 잘라서 사용해야 한다. 이렇게 낱장 자동공급장치가 없는 프린터를 사용하는데 낱장용지에 인쇄해야 하는 경우에는 인쇄메뉴의 W(Wait) 항목을 Y로 지정하고 인쇄를 시작하면 매 페이지마다 인쇄가 잠시 중단되어 낱장용지를 사용하여 인쇄할 수 있게 해준다.

dBASEⅢ+ 화일의 이용

이상으로 간단한 예와 함께 팔란티어 워드프로세서의 발송문 기능을 알아 보았다. 위의 예에서는 자료화일을 직접 팔란티어내에서 작성을 했는데, 만약 다른 프로그램에서 작성된 데이터가 있다면 팔란티어 내에서 다시 작성할 필요없이 그 데이터를 그대로 이용할 수 있다.

팔란티어 워드프로세서의 발송문 기능은 자체내에서 작성한 자료 화일뿐 아니라 SDF 형식으로 되어 있는 외부자료화일도 사용할 수 있도록 되어 있기 때문이다. SDF 형식이란 각 필드는 입력된 데이터의 길이에 관계없이 미리 정해진 길이를 갖고, 각 레코드는 1행 단위로 구분되어 있는 자료의 형태를 의미한다. 그런데 데이터 관리를 위한 프로그램에서 작성된 데이터는 그들 나름의 독특한 형태를 가지고 있다. 그러므로 발송문 화일에서 그 데이터를 사용하려면 먼저 SDF 형식으로 변경해야 한다.

그럼 dBASEⅢ+에서 작성된 CUSTOMER.DBF란 화일을 발송문의 자료화일로 사용하는 방법을 알아보자.

CUSTOMER.DBF의 구조와 내용은 **화면 4**와 같다. 이렇게 만들어진 DBF화일을 SDF형식의 화일로 변경하려면 dBASE Ⅲ+의 DOT PROMPT 상태에서 아래와 같이 입력한다.

```
. USE CUSTOMER        ←COSTOMER. DBF화일을 OPEN한다.
. COPY TO CUSTOMER SDF   ←SDF형식으로 COSTOMER.
                           TXT라는 화일에 복사하라.
```

이 결과 CUSTOMER.TXT라는 SDF형식의 자료 화일이 만들어 진다. CUSTOMER.TXT의 내용은 **화면 5**와 같다. **화면 6**이 CUSTOMER.TXT를 이용하여 발송문 화일을 인쇄하기 위한 문서이다. 문서의 상단 화일과 변수를 정의해주는 부분이 앞에서 소개된 발송문의 내용과 조금 다르다.

1) 인용부호 안은 자료화일 이름이고, 괄호 안의 숫자는 1레코드의 문자수이다. 1레코드에 해당하는 문자수는 필드의 합에 2를 더한 값이다.
2) 해당 필드를 읽어들일 변수명과 함께 그 필드의 폭을 써준다.

이상으로 팔란티어 워드프로세서 내의 발송문 기능을 이용하여 다량의 우편물 인쇄를 효과적으로 처리하는 과정과 다른 프로그램에서 작성된 데이터를 사용하는 방법을 소개해 보았다. 이 기회를 통해 발송문 기능의 확실한 이해와 앞으로 실제 업무에서 잘 활용할 수 있기를 바란다. PC

〔화면 4〕 COUSTOMER.DBF의 구조

```
. list structure
데이타베이스의 구조: A:customer.dbf
레코드의 수        :     4
최종 수정일        : 01/01/80
 필드  필드명         형식        폭 소숫점
   1  CARDNO        숫자식        3
   2  NAME          문자형       10
   3  COMPANY       문자형       20
   4  ADDRESS       문자형       30
   5  ZIP           문자형        7
   6  TEL           문자형        8
   7  DATE          날짜형        8
** 합계 **                      87

. list
레코드# CARDNO NAME   COMPANY          ADDRESS                 ZIP     TEL      DATE
     1      1 김 영한  대한 소프트웨어    영등포구 여의도동 30-17     150-010 783-0201 10/15/88
     2      2 강 현숙  서진상사(주)       강남구 논현동 365-10       135-010 345-0102 01/25/89
     3      3 서 동찬                   영등포구 여의도동 ABC아파트  150-010 785-0203 02/11/89
     4      4 한 만구  세계소프트웨어     마포구 공덕동 170번지       121-020 719-1617 03/30/89
```

〔화면 5〕 SDF의 자료형식

```
1김 영한   대한 소프트웨어   영등포구 여의도동 30-17      150-010783-020119881015
2강 현숙   서진상사(주)      강남구 논현동 365-10        135-010345-010219890125
3서 동찬                    영등포구 여의도동 ABC아파트   150-010785-020319890211
4한 만구   세계소프트웨어    마포구 공덕동 170번지        121-020719-161719890330
```

〔화면 6〕 문서화일의 출력양식 지정화면

```
CFILE "CUSTOMER.TXT" F(88)
CREAD NO(3), NAME(10), COMPANY(20), ADDRESS(30), ZIP(7), TEL(8), DATE(8)


\COMPANY
\NAME 님께


     폐사에서 주최하는  소프트웨어 전시회에 \NAME 님을
     초대합니다.

     오셔서 자리를 빛내 주시기 바랍니다.
```

```
150-010

영등포구 여의도동 36-2
대성 소프트웨어(주)

                              150-010
                              영등포구 여의도동 ABC아파트
                              김 정훈 님 귀하
```

```
김 정훈 님께

대성 소프트웨어(주)에서는 팔란티어 워드프로세서 교육
수강자를 대상으로 3월 10일 오후 4시 특강을 실시합니다.
관심있는 분들의 많은 참석을 바랍니다.
```

〈문서화일의 출력결과〉

명령어 구성형식

김용득/아주대 전자과 교수

명령어 구성형식을 언급하기전에, 먼저 각각의 정보가 어떠한 방법으로 레지스터에 저장되는지, 이런 여러 방법들을 알아보는게 순서일성 싶다. 이번 달에는 주소지정방식의 예를 설명하고 명령어 구성형식에 대해 알아보고자 한다.

지금까지 연재된 하드웨어강좌 (1), (2)부에서는 간략하게나마 마이크로프로세서의 구성 및 설계방법에 관하여 설명하였다. 이번회까지는 마이크로프로세서 시스템의 활용과 설계에 있어 기초적인 여러 문제들을 초보자가 이해할 수 있도록 가능한 쉽게 설명하였지만 전달과정에서 무리(?)가 있었다면 좀더 많은 관심을 갖고 의문스러운 점을 하나씩 풀어나가는 자세로 학습하면 많은 도움이 되리라 생각된다.

특히, 다음호부터는 이제까지 설명한 것들을 바탕으로한 마이크로프로세서의 활용 및 응용, 시스템 설계 등의 실질적인 문제와 접하게 되므로 지금까지의 내용을 충분히 이해하고 시작하면 어렵게만 느껴지는 마이크로프로세서에 대한 문제를 해결할 수 있으리라 믿는다. 그럼, 본론으로 들어가 주소지정방식부터 설명하겠다.

주소지정방식

컴퓨터에서 사용되는 모든 명령어의 구성형식은 무엇을 처리할 것인가의 기능을 정의하는 OP코드와 OP코드부에 주어진 기능을 처리할 때 필요로 하는 정보들의 지정으로 다음과 같이 구성된다.

OP코드부	정보값	· · ·	정보값

이때 OP코드부는 보편적으로 8비트 (256개의 서로 다른 명령어를 지정할 수 있음.)로 구성되며 정보값은 처리될 내용이 무엇이냐에 따라 1개 또는 1개 이상의 정보지정값을 갖게 된다. 이들은 보통 기억장치에 저장되어 있거나, 마이크로프로세서의 내부레지스터에 있거나, 또는 처리될 정보를 직접 지정하여야 하며, 각각의 비트수도 지정방식에 따라 변화된다. 그러므로 어느 마이크로프로세서이든지 각각의 특성에 따라 효율적으로 이들 정보를 표출하여야 하며, 이러한 방법을 주소지정방식(Addressing Mode)이라 한다.

8086 또는 80286 마이크로프로세서에서는 이들 정보처리방식을 처리되는 기능에 따라 다음 4가지 방식중의 하나로 분류한다.

① 내부레지스터 사이의 정보처리방식(Resister to Register)
② 직접적인 정보값과 내부레지스터와의 정보처리방식(Immediate to Register)
③ 내부레지스터와 주기억장치에 저장된 정보의 처리방식(Register to Memory or Memory to Regsiter)
④ 직접적인 정보값과 주기억장치에 저장된 정보와의 처리방식(Immediate to Memory)

특히, 이들중 주기억장치를 사용할 때 실질적인 주소 지정방식은 그림 1과 같이 나타낼 수 있으며, 처리할 정보가 저장된 주기억장치내의 실제주소는 이들 각각의 방식을 선택적으로 조합하여 다음과 같이 다양하게 구성할 수 있다.

주소지정을 위해서는 '명령어가 기억된 주소를 지정할 것인가?' 또는 '처리될 정보가 저장된 주소번지인가'에 따라 다르므로, 명령의 처리내용과 주소의 지정방식에 따라 다음과 같이 9가지로 분류하여 설명하겠다.

● 직접 정보값을 지정하는 모드

OP코드부	모드지정부	정보값
8 비트	8 비트	8 비트 또는 16비트

이는 명령의 정보값을 8비트 또는 16비트로 직접 지정하는 모드이다. 여기서 세부적인 모드지정부는 명령어 구성형식에서 설명하기로 하고, 직접 정보값을 지정하는 모드(Immediate Addressing)에서는 모드지정부 다음에 연속하여 처리될 정보값이 1바이트 또는 2바이트로 지정됨을 표시하게 된다. 따라서 처리될 정보값이 명령어 내부(3 또는 4바이트)에 포함되어 있다.

이 모드와 뒤에 언급할 레지스터에 정보값이 지정된 모드는 CPU내부의 데이터를 정보값으로 사용하기 때문에 메모리중의 정보값을 참조하는 경우보다 고속으로 명령을 처리할 수 있다.

● 직접 주소를 지정하는 모드

OP코드부	모드지정부	주소지정값
8 비트	8 비트	16비트

직접 주소를 지정하는 모드(Direct Addressing)에서 모드지정부 다음의 2바이트는 현재 처리되는 세그먼트 레지스터(DS)와 함께 정보가 저장된 주기억장소를 지정한다. 즉 실제 기억장치의 주소는 DS×16+주소지정번지의 합으로 20비트가 할당된다.

●레지스터에 정보값이 저장된 모드

8비트	8비트
OP코드부	모드지정부

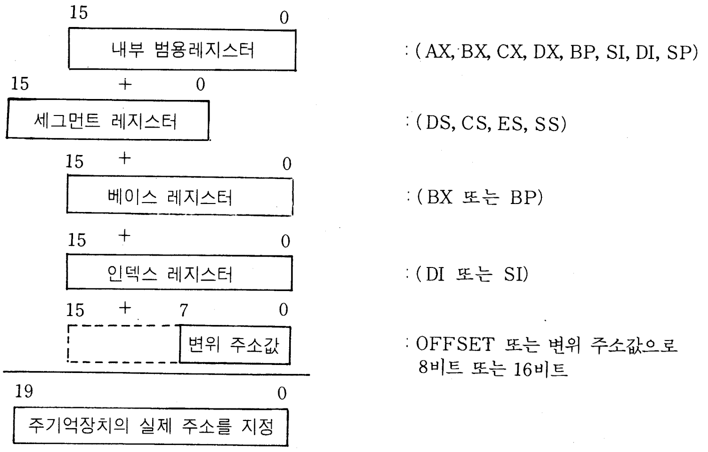
레지스터에 정보값이 저장된 모드(Register Addressing)에서 정보는 범용레지스터에 기억되어 있으며, 모드지정부에서 어느 레지스터에 처리될 정보가 있는가를 지시하고 있다. 따라서 이 모드에서는 명령어의 길이가 2바이트로 지정되는 장점이 있다

●레지스터에 의한 간접 주소지정 모드

8비트	8비트
OP코드부	모드지정부

레지스터에 의한 간접적으로 주소를 지정하는 모드(Register Indirect Addressing) 방식은 레지스터에 정보값이 저장된 모드방식과 같은 명령어 구성체계이나, 이는 처리될 정보값이 주기억장치에 있으며, 이의 주소번지를 모드지정부에서 지시하는 레지스터에 16비트 주소값이 있게 하고, 세그먼트 레지스터(DS)와 앞의 방식과 같이 결합되어 20비트의 실제 주소를 지정하게 된다.

〔그림 1〕 주기억장치의 실제 주소지정방식



●인덱스 레지스터를 사용한 주소지정모드

8비트	8비트	8비트 또는 16비트
OP코드부	모드지정부	주소지정값

모드 지정방식에서 인덱스 레지스터라 부르는 내부레지스터(주로 BX, BP, SI, DI사용)가 지정된다. 실제 주소번지는 내부레지스터값과 주소지정값을 합한 주소가 된다. 이러한 인덱스 레지스터를 사용한 주소지정모드(Indexed Addressing)는 블록정보의 특정화일을 호출하기에 편리한 방식이다.

즉, 주소지정값을 일정하게 두고 인덱스 레지스터값을 증감시켜 매 특수화일의 처리를 쉽게 하도록 한 것이다.

●베이스 레지스터를 사용한 주소지정모드

8비트	8비트	8비트 또는 16비트
OP코드부	모드지정부	주소지정값

베이스 레지스터를 사용한 주소지정 모드(Based Addressing)는 인덱스 레지스터를 사용한 주소지정방식과 똑같이 운영되나 단지 차이점은 베이스 레지스터값은 일정하게 두고 주소지정값을 변화시키면서 필요한 정보를 지정하는 방식이다.

●베이스와 인덱스 레지스터를 사용한 주소지정모드

8비트	8비트	8비트 또는 16비트
OP코드부	모드지정부	주소지정값

말 그대로 인덱스 레지스터와 베이스 레지스터를 사용한 주소지정모드(Based Indexed Addressing)가 결합된 형태로 베이스 레지스터(BX 또는 BP사용)와 인덱스 레지스터(SI 또는 DI사용)의 값과 주소지정값의 합이 실제 주소번지값이 되며, 임의의 길이를 갖는 정보의 순차처리방식에 유용하게 쓰이고 있다.

●문자처리를 위한 주소지정모드

8비트
OP코드부

1바이트로 구성되어 있으며 문서처리 등과 같이 문자정보의 고속처리(이동)를 위한 응용으로써 이동주소번지는 SI레지스터(이동 할 위치)와 DI레지스터(이동 할 위치)를 지정하게 되며, 이동될 때까지 자동으로 주소값이 증감되도록 되어 있다.

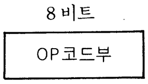
●입/출력장치 지정모드

정보를 읽거나 쓸 때•외부의 입출력장치를 지정하는 모드이며, 동작방식에 따라 직접 입출력 지정방식과 간접입출력 지정방식으로 분류된다.

8비트	8비트
OP코드부	직접모드지정부

직접 입출력 지정방식은 256개의 입력장치와 출력장치를 각각 지정하는

방식으로 입출력 포트지정(8비트)을 OP코드부 다음에 직접 지정하는 것이다.



또한, 간접 입출력 지정방식은 64K개의 입출력장치를 각각 부착시킬 수 있고, 이를 지정하기 위하여 먼저 내부레지스터(DX 레지스터를 사용함)에 입출력장치의 지정값을 넣고 1바이트의 OP코드로 호출한다. 이때 정보의 전송은 8비트 또는 16비트 정보에 따라 반드시 지정된 입출력장치와 AL, AX 내부레지스터를 사용하며, 세부적인 내용은 다음달에 입출력장치를 설명한 후, 이들의 하드웨어와 동작운영을 설명하도록 하겠다.

다음에 처리될 명령어의 위치를 지정하기 위한 주소지정모드는 세그먼트 레지스터(CS)와 명령어지정 레지스터(IP) 모두를 변경해야 하나, 모든 프로그램의 기억된 상태는 기억장치의 효율적 운영을 위해 64KB 단위 기억소자 블록 즉, 세그먼트 단위로 운영하고 있기 때문에(이 부분은 다음의 기억소자 운영부분에서 자세히 논하려 함.) 같은 세그먼트내에서의 점프이동은 세그먼트 레지스터를 변경하지 않고 직접 명령어지정 레지스터(IP)값만 지정하고도 이동이 가능하다. 즉 이러한 방식을 내부세그먼트 브랜치방식이라 부르며, 명령어내의 주소지정값에 따라 현재의 명령어 위치로부터 128개의 주소를 가감하여 점프되는 직접방식과 앞에서 설명된 주소지정모드에 따라 점프될 주소를 계산하여 지정하는 간접방식으로 나눌 수 있다.

다른 세그먼트에 있는 프로그램으로 점프되는 경우(Inter Segment)는 CS와 IP를 모두 변경해줘야 하며, 여기에도 세그먼트와 명령어지정 레지스터를 어떻게 지정하느냐에 따라 직접방식과 간접방식으로 나누어지고 위에서 설명한 방식과 같이 결정된다.

명령어의 구성형식

8086이나 80286에서 사용되는 명령에는 그림 2와 같이 OP코드부만 있는 1바이트 명령어로 부터 6바이트 명령어 등의 다양한 명령어 길이를 갖고 있으며, 이는 처리할 정보지정방식에 따라 변환된다. 어떤 명령어든지 첫번째 바이트는 OP코드부이며, 직접 주소지정을 위한 2바이트, 변위값은 1 또는 2바이트, 세그먼트 주소값은 2바이트 등으로 연속된다. 이때 이들 주소번지나 정보값을 직접 지정해줄 경우에는 하위순위의 값들이 먼저 나오게 된다 **(그림 2 참조)**.

그림 2에서 OP코드부는 무엇을 처리할 것인가의 동작지정방식과 특수처리용 비트가 포함되는데, 특수처리용 비트에는 W와 S비트의 의미가 복합적으로 사용된다.

W비트는 다음에 처리될 정보값이 8비트(W=0)인가 16비트(W=1)인가를 지정하기 위한 것이고, S(부호)비트는 사용될 정보의 음수(S=1)/양수(S=0)를 표시하는 것으로 연산처리인 경우에만 사용된다.

또한, OP코드는 명령어에 따라 그림 2와 같이 2바이트가 사용될 수도 있는데 두번째 OP코드부는 반드시 그림 3(a)의 형식을 갖게 된다**(그림 3 참조)**.

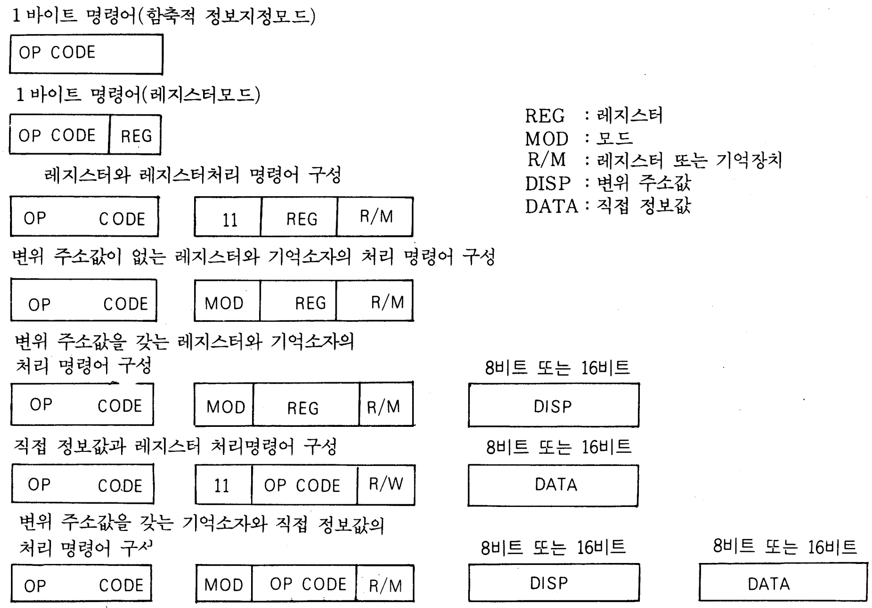
그리고 모드부와 R/M부(레지스터 또는 기억소자 지정모드)는 그림 3(a)와 같이 처리될 정보값이 주소지정방식에서 설명된 것과 같은 어떤 형태로 지정되는가를 보여주고 있다.

예를 들어 R/M가 "001"이고 모드부가 "01"이면, 정보값의 실제주소번지는 베이스와 인덱스 레지스터를 사용할 주소지정모드 "(BX)+(DI)+(D8)로 계산된다. 그러나 내부레지스터에 정보값이 있는 경우에는 내부레지스터 주소가 그림 3(a)와 같이 정의되며, 이 코드를 직접 사용함으로써 지정할 수 있다.

ADD(덧셈) 명령어가 어떻게 구성되며 활용되는가를 다음과 같이 예를 들어 설명하겠다.

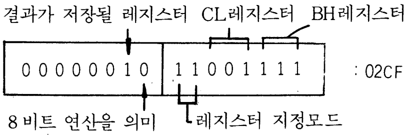
즉, 덧셈을 수행하기 위한 명령어 구성형식은 **그림 4**와 같은 구성형식을 가지나 이의 활용은 레지스터와 레지스터의 연산, 기억소자와 레지스터의 연산, 직접정보와 레지스터 또는 기억소

〔그림 2〕 명령어 구성형식

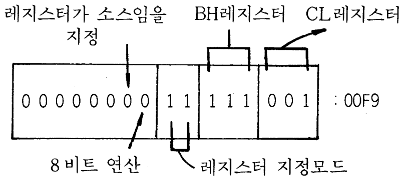


자와의 연산으로 구분되며, 연산결과의 기억방식에 따라서 또는 레지스터의 종류에 따라 여러 방식으로 구성시킬 수 있다.

그 일예로써 BH와 CL레지스터값을 더하여 CL레지스터에 저장하기 위한 명령어 형식은

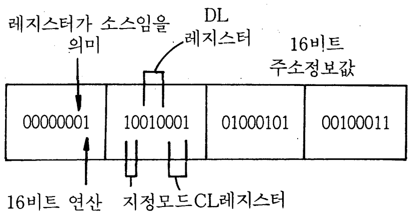


또는,



가 되며 "00CF와 00F9"과 같은 이들 두 명령어는 "CL←(BH)+(CL)"을 수행하는 명령어임을 뜻한다.

독자들의 이해를 돕기 위하여 베이스 레지스터와 인덱스 레지스터를 사용한 지정된 기억소자의 정보값과 레지스터와의 덧셈의 예를 하나 더 설명하겠다.



실제주소번지=(BX)+(DI)+(D16)

이는 4바이트 명령어를 형성하며, 기계어 명령어는 "01914523"으로써 만약 BX레지스터에 "0892", DI레지스터에 "59A3"가 기억되어 있다고 하면, 주기억장치의 실제 주소번지는 "0892+59A3+2345=857A" 가 된다. 이는 현재 세그먼트내의 857A번지에 연산될 정보가 있음을 의미한다.

이와 같이 덧셈을 위해서도 다양한 명령어 구성이 가능하며, 기계어 번역과정은 시스템 프로그램의 번역 여부에서 변환과정이 자세히 설명된다.

명령어 수행시간

하나의 명령어를 읽어서 처리하는데 소요되는 시간을 명령어 수행시간이라 하며, 이는 마이크로프로세서의 클럭 입력주파수에 따라서 결정되나 이미 주어진 명령에 따라 요구되는 시스템 클럭수가 결정되어 있으므로 이의 결정방법을 설명하겠다.

실제로 하드웨어에 관련된 제어응용에서는 명령어 수행시간이 매우 중요하다.

여기서는 "이들이 어떻게 사용되는가?"에 대한 의문을 명령어 종류를 설명한 후 주변 하드웨어를 설계할 때 예를 들어 설명하겠다.

표 1에는 덧셈 명령어에서 명령어 수행시간을 보여주며, 표 2에서는 명령어 형식이 주소지정모드에 따라 주소계산시간이 다르게 나타나며 이의 시간을 보여준다.

이들 시간은 마이크로프로세서가 설계될 때 결정된 값으로 우리는 이들 결과값을 사용하여 주변 응용 하드웨어를 설계하게 된다. 예를 들어 클럭발생기에 입력되는 수정발진자의 클럭이 15MHz라 하면 이 시스템 시스템클럭은 5MHz 즉, 0.2μs의 클럭주기가 되므로 레지스터 내용과 다른 레지스터와의 덧셈인 경우 표 1에서 보여준 바와 같이 3개의 클럭주기가 필요하다. 따라

〔그림 3〕 주소지정모드의 표시방식

2	3	3
모드부	확장 OP 코드부	R/M

2	3	3
모드부	내부레지스터	R/M

(a) 2바이트 OP코드형식

내부레지스터 주소	레지스터 이름	
	W=1	W=0
0 0 0	AX	AL
0 0 1	CX	CL
0 1 0	DX	DL
0 1 1	BX	BL
1 0 0	SP	AH
1 0 1	BP	CH
1 1 0	SI	DH
1 1 1	DI	BH

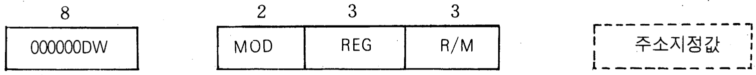
(b) 내부레지스터의 주소지정번지

MOD / R/M	0 0	0 1	1 0	1 1	
				W=0	W=1
0 0 0 (세그먼트 이름)	(BX)+(SI) DS	(BX)+(SI)+D8 DS	(BX)+(SI)+D16 DS	AL	AX
0 0 1 (세그먼트 이름)	(BX)+(DI) DS	(BX)+(DI)+D8 DS	(BX)+(DI)+D16 DS	CL	CX
0 1 0 (세그먼트 이름)	(BP)+(SI) SS	(BP)+(SI)+D8 SS	(BP)+(SI)+D16 SS	DL	DX
0 1 1 (세그먼트 이름)	(BP)+(DI) SS	(BP)+(DI)+D8 SS	(BP)+(DI)+D16 SS	BL	BX
1 0 0 (세그먼트 이름)	(SI) DS	(SI)+D8 DS	(SI)+D16 DS	AH	SP
1 0 1 (세그먼트 이름)	(DI) DS	(DI)+D8 DS	(DI)+D16 DS	CH	BP
1 1 0 (세그먼트 이름)	D16 DS	(BP)+D8 SS	(BP)+D16 SS	DH	SI
1 1 1 (세그먼트 이름)	(BX) DS	(BX)+D8 DS	(BX)+D16 DS	BH	DI

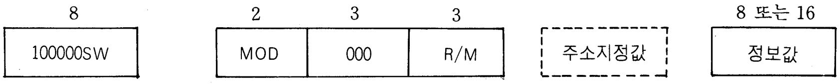
D8 : 8비트 변위 주소값　　D16 : 16비트 변위 주소값

(c) 모드부와 R/M영역의 정의

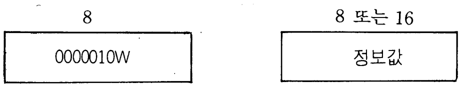
〔그림 4〕 ADD 명령어의 구성형식



(a) 레지스터와 레지스터 또는 기억소자속의 정보값과의 덧셈



(b) 정보값과 레지스터 또는 기억소자속의 정보값과의 덧셈



(*여기서 주소지정값은 모드에 따라 선택적으로 결정되며 정보값은 W비트에 의하여 8 또는 16비트가 지정됨.)

(c) AX 또는 AL과 정보값과의 덧셈

〔표 1〕 명령어 수행시간

명령어형식	필요한 클럭주기의 수
레지스터와 레지스터	3
기억소자와 레지스터 연산후 레지스터에 결과기록	9+주소계산시간
레지스터와 기억소자와의 연산후 기억소자에 결과기록	16+주소계산시간
직접 정보값과 레지스터 연산후 레지스터에 결과기록	4
직접 정보값과 기억소자와의 연산후 기억소자에 결과기록	17+주소계산시간

〔표 2〕 주소 계산시간

주소지정방식	필요한 클럭주기의 수
직접 주소지정모드	6
간접 레지스터 주소지정모드	5
연관 레지스터 주소지정모드	9
베이스와 인덱스 레지스터를 사용한 주소지정모드	
(BP)+(DI) 또는 (BX)+(SI)	7
(BP)+(SI) 또는 (BX)+(DI)	8
(BP)+(DI)+주소지정값	11
(BP)+(SI)+주소지정값	12

서 이 명령어 수행시간은 0.6㎲가 된다.

좀더 복잡한 예로 레지스터 내용과 기억소자의 내용을 더하여 결과를 기억소자에 저장한다. 이때의 주소지정모드를 베이스 레지스터와 인덱스 레지스터를 사용하고자 하면 명령어 수행시간은 16+8 즉 24개의 주기가 필요하며 명령어 수행시간은 4.8㎲가 된다. 그러나 이들은 처리될 정보의 위치가 짝수번지에 있는 경우이며, 홀수인 경우에는 4개의 추가된 클럭이 더 요구되므로 16+8+4+4 즉 32개의 주기로써 6.4㎲가 걸린다. 따라서 실제 응용에서는 어느 명령어를 적용하느냐에 따라, 기억된 정보의 주소값이 홀수냐 짝수냐에 따라 모두 다르기 때문에 하드웨어에 관련된 제어신호 구성방법은 다양하게 이뤄진다.

μP의 명령어

8086에 사용되는 명령어의 총수는 약 300여개 정도이나 이를 기능별로 구분하면 다음과 같다.

① 정보의 이동을 위한 명령어(Data Transfer)
② 연산처리를 위한 명령어(Arithmetic)
③ 비트처리를 위한 명령어(Bit Manipulation)
④ 문자처리를 위한 명령어(String Manipulation)
⑤ 명령어 처리순서 등의 제어를 위한 명령어(Control Transfer)
⑥ 프로세서 제어를 위한 명령어(Process Control)

비록 300여개의 명령어를 갖고는 있으나 모두 상이한 것이 아니고 유사한 종류 몇개로 구분되므로 독자는 보편적인 기초가 되는 명령어(이를 어셈블리어라 함.)를 사용하여 프로그램을 숙지하길 바란다. 즉 정보이동을 위한 명령어중에 MOV란 명령어는 같은 형식을 사용하여 28개의 서로 다른 기계어 형식을 갖고 있다.

정보의 이동을 위한 명령어에는 일반적인 정보들의 이동을 위한 명령어, 입출력장치와의 정보전송을 위한 명령어, 주소이동을 위한 명령어 및 내부플래그 처리를 위한 명령어들로 나누어지며, 매뉴얼이나 프로그램 참고자료에 이들 명령어가 도시되어 있으므로 세부적인 내용의 설명은 생략하겠다.

연산처리를 위한 명령어는 가·감·승·제를 처리하기 위한 명령어로 수의 개념에 따라 2진수, 10진수 등의 여러 형태로 분류되어 사용된다. 특히 연산처리 명령어는 정보의 형태를 변형시키는 명령어이므로 반드시 변형된 형태를 지시하는 플래그 레지스터를 검사 및 확인을 하여야 한다.

비트처리를 위한 명령어는 논리연산과 시프트 동작 등과 같이 이웃된 비트와는 무관하게 독립적으로 클럭비트를 처리하는 것이고, 문자처리를 위한 명령어는 문자편집기(Word Editor)와 같이 고속으로 문자이동 처리를 위한

것이다.

명령어 처리순서를 제어하기 위한 명령어는 서브루틴의 처리, 점프의 반복 계산 및 인터럽트처리 등과 같이 일정한 프로그램 순서를 변경하여 처리하고자 할 때 사용한다.

끝으로 프로세서 내부상태를 제어하는 명령어로는 플래그 처리, 정지(HALT)와 대기(WAIT) 등과 같은 외부 소자와의 동기신호 발생을 위한 하드웨어와 관련된 기계어 처리를 위한 명령어를 들 수 있다. 이와같은 명령어중 시스템 활용을 위한 예제와 중요한 명령어는 다음호에 계속 설명하겠다. PC

● 디지탈 강좌(6) ●

메모리와 보조기억장치

김정선/한국항공대학 전자과 교수

IC 메모리

IC 램에는 지난 4월호에 설명한 스태틱 램외에 다이나믹 램이 있다. 스태틱 램은 한번 기억시킨 내용은 전원을 끄거나 다른 내용으로 기억시키기 전에는 바뀌지 않는 반면 다이나믹 램은 기억시킨 내용이 시간이 지나면서 점차 소멸되어 일정한 시간간격으로 재기억시키지 않으면 그 내용을 보존할 수 없게 된다. 기억내용을 계속 보존하기 위해 일정한 시간간격으로 재기억시키는 것을 리플레시(Reflesh)라고 한다.

다이나믹 램의 원리는 다음과 같은 예를 통해 좀더 이해를 도울 수 있을 것이다.

대단히 긴 동축선의 한쪽 끝에서 펄스를 가하면 다른쪽 끝으로 펄스파형이 전달된다. 이 전달과정에서 선로의 전달지연 특성 등으로 인하여 입력단의 펄스파형과 출력단의 펄스파형 사이에는 시간적 지연과 진폭의 감쇄가 발생할 것이라는 점을 추측할 수 있다 (그림 1(a), (b) 참조).

그림 1(c)와 같이 입력스위치를 1의 위치에 놓고 펄스를 가한 순간 입력스위치를 2의 위치로 옮겼을 경우 시간지연과 진폭이 감쇄된 출력이 발생된다. 그러나 이 출력은 증폭기를 통하여 감쇄된 만큼 증폭되어 다시 입력에 가해진다. 이와 같이 재입력된 펄스는 또 다시 시간지연과 감쇄된 진폭이 출력측에 나타나고, 다시 증폭되어 입력측에 가해진다.

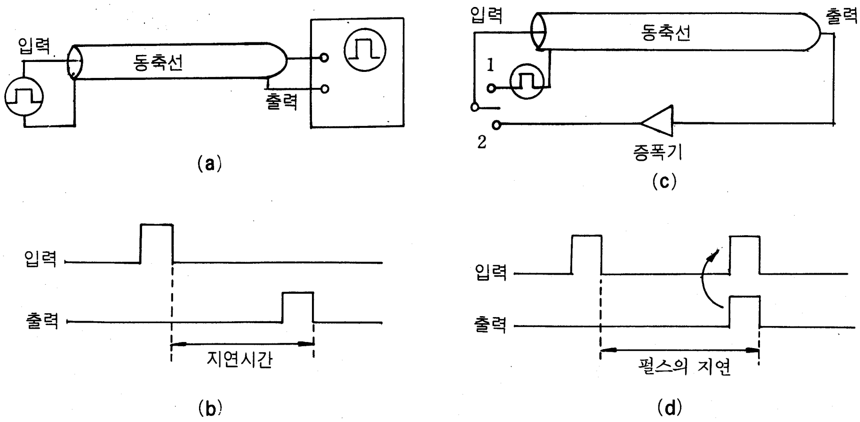
이와 반면에 스위치를 2의 위치에 고정시켜 두었을 경우 앞서와 같은 동작이 반복되며, 동축선상에는 펄스 파형이 처음에 가한 형태대로 항상 존재하게 된다. 이러한 현상 역시 기억의 일종이다.

그러나 출력에서 입력측으로 증폭기를 통한 궤환회로가 없다면, 펄스파형의 기억은 펄스가 동축선을 전파하는데 걸리는(지연시간 동안) 시간동안만 존재하며 이시간 만큼만 기억이 가능한 것이다. 여기서 궤환회로는 리플레시회로이며 증폭기의 출력신호는 리플레시신호가 되고, 지연특성을 이용한 동축선은 다이나믹 메모리소자가 되는 셈이다.

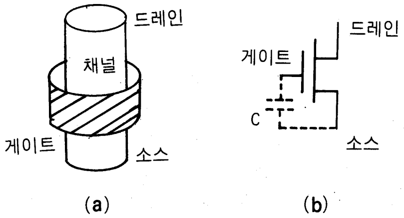
MOSFET는 그림 2에 보인 바와 같이 채널과 게이트로 구성되며, 게이트의 전압에 따른 전계에 의하여 소스-드레인 전류가 제어되고 스위치 동작이 가능하다. 채널에 전계를 주기 위한 게이트와 채널은 일종의 커패시터(용량) 역할도 한다. 따라서 게이트와 채널로 형성되는 용량사이에 전압을 가하였을 때 충전이 될 수 있다. 이렇게 충전된 전압은 어느 정도의 시간동안 유지되며 주변회로 혹은 게이트와 채널사이의 절연정도 등에 의하여 서서히 방전한다.

게이트에 가한 전압에 의하여 소스-드레인간에는 스위치 동작이 이루어지며 한번 가한 게이트전압이 충전되고, 방전하면서 영(0)전압이 될때 까지 스위치 동작은 지속된다. 즉 게이트에 전압을 가하여 스위치 동작이 되도록하는 동작을 유지하는 기억상태가

〔그림 1〕 동축선 기억원리



〔그림 2〕 MOSFET



된다. 그러나 그 기억상태는 게이트에 전압이 충전되어 있는 동안만 유지된다. 따라서 기억상태를 오래도록 보존하려면 충전된 전압이 모두 방전하기 전에 재충전을 시켜야만 한다. 즉 리플레시를 시켜야 한다. 이러한 원리를 이용한 기억장치가 IC다이나믹 램이다.

이런 다이나믹 램은 리플레시회로의 구성 등 회로구성상 이용에 불편한 점은 있으나 여러개의 트랜지스터 및 저항소자들로 구성되는 스태틱 메모리에 비하여 한개의 트랜지스터로 1비트의 메모리소자 구성이 가능하며, IC화하기에 용이하고 제작단가를 저렴하게 할 수 있고, IC의 메모리 집적도를 높일 수가 있어 현재 많이 사용되고 있는 기억소자형이다. 특히 IC기술의 발달로 이용자에게 불편한 리플레시 회로를 IC 메모리내에 내장시켜 사용자 입장에서는 마치 스태틱 램과 같이 사용할 수 있도록 한 것도 있다.

ROM과 PROM

ROM(Read Only Memory)은 읽기만 가능한 즉 써넣는 것이 불가능한 기억장치이다. 다시 말해서 ROM에 저장되는 2진 정보는 생산시에 하드웨어적으로 고정시켜서 다른 내용의 정보를 기록하지 못하게 되어 있다.

ROM은 입력의 조합에 대해 일정 출력조합을 얻어내는 회로로서 조합논리회로의 일종이다. 여기서 입력조합은 어드레스 입력에 해당하며, 출력조합이란 기억된 정보의 출력이다.

ROM의 이용은 주로 코드 변환기 등에 쓰이며 2진 코드의 그레이 코드 변환, 문자코드의 발생 등이 있다.

ROM의 구성과 동작은 다음 예로서 설명할 수 있다. 즉 8비트로 구성되는 4개의 워드 기억회로 예를 **그림 3**에 보였다.

그림 3에서 A_1A_0는 어드레스 입력으로, 이 A_1A_0의 입력조합은 4개의 AND 게이트로 구성된 어드레스 디코더에 가해지며 이 어드레스 디코더의 출력은 워드선을 선택하게 된다. 선택된 워드선에 다이오드로 연결된 열선은 비트출력을 "1"로, 연결되지 않은 열선은 비트출력을 "0"으로 한다.

그림 3에서 A_1A_0=0, 0 일때 출력은 10101010이고, A_1A=0, 1 일때 출력은 11001100이다. 열선의 수에 따라 한 워드를 구성하는 비트수가 달라질 수 있으며, 워드선의 증가에 따라 기억 워드를 증가시킬 수 있다. 이때 워드선을 많게 하려면 이를 충분히 선택할 수 있도록 어드레스 입력선도 많아져야 한다. 어드레스 입력선과 워드선 사이에는 어드레스선 수를 n, 워드 수를 N이라 할때

$$2^n = N$$

의 관계를 갖는다. 예를 들어 어드레스선이 10개이면 워드수는 1024개까지 가능하다.

이런 형의 ROM은 사용자에 의하여 기억내용을 변형시킬 수 없다. 그러나 단 한번 필요한 정보를 사용자가 기억시킬 수 있는 PROM이 있는데 이는 다른 내용으로의 변경은 불가능하다.

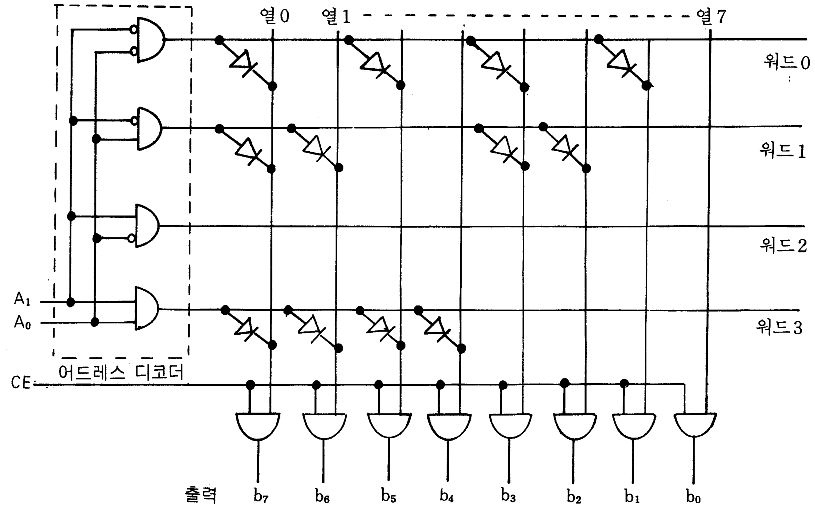
그림 4의 회로가 그림 3의 ROM회로와 다른점은 제작회사에서 출고될 때 모든 열과 워드선 사이에 휴즈와 직렬로 다이오드가 연결된 상태로 출고되며, 사용자의 필요에 따라 휴즈를 절단하므로서 필요한 내용의 기록이 가능하다는 점이다. 한번 절단된 휴즈는 다시 연결할 수 없으므로 기억내용의 변경은 불가능한 것이다. 이런 형의 메모리를 휴즈형 PROM(Programmable ROM)이라고 한다.

EPROM과 EAROM

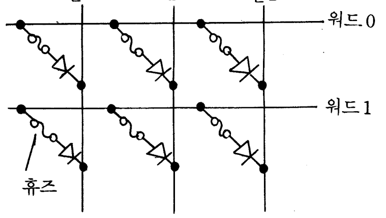
ROM과 PROM은 모두 그 내용을 한번 밖에 프로그램할 수 없는 특성을 갖는다. 그러나 ROM은 앞에서 설명한 것과 또 다른 두가지 형태의 EPROM (Erasable PROM)과 EAROM(Electrically Alterably ROM) 등의 비휘발성 특징을 갖는 것이 있다. 이는 기록된 정보를 임의로 변경시킬 수 있다.

물론 이런 비휘발성 ROM은 동작전

〔그림 3〕 ROM 회로예



〔그림 4〕 휴즈형 PROM



원에 무관하게 그 내용은 보존되어 한번 기록된 정보는 그 내용을 기록방식에 의하여 변경시키기 전에는 변환되지 않는다.

이러한 특성을 갖는 EPROM은 그 저장내용을 변경하고자 할 때는 부분적인 수정은 불가능하며 메모리 칩에 준비되어 있는 윈도우(빛을 쬐일 수 있는 창)를 통하여 자외선을 쬐일 경우 전에 기억되었던 정보는 모두 지워진다. 이렇게 동시에 전내용을 지운 후 새로운 내용의 정보를 기록한다.

이와 같이 자외선에 의하여 메모리의 기억정보를 동시에 모두 지우고 다시 프로그램할 수 있는 EPROM을 UVEPROM (Ultra Violet EPROM)이라 하며 1971년 인텔사에 의해 처음으로 소개되었고 FAMOS(Floating Avalanche Injection MOS) 집적화를 실현하였다. FAMOS EPROM(혹은 UVEPROM)의 액세스시간은 대략 450nsec~1μsec 정도이며, 칩상단의 투명한 유리창을 통하여 파장이 2537Å인 자외선을 6~10Watt-sec/㎠의 세기로 비출 때 소거시간은 10~30분 정도 걸린다.

이 EPROM을 프로그램하기 위하여는 회로에 연결한 상태에서는 할수 없고 회로에서 분리시켜 자외선에 의해 기억내용을 소거시킨 다음 EPROM 프로그래머로 새로운 내용을 기억시켜야 한다. 라이트시간은 수 msec~30분 정도이다.

앞에 설명한 EPROM은 기억내용을 변경하는 방법이나, 절차가 다소 복잡하여 프로그래머나 자외선 발생기 등의 기구와 많은 시간을 필요로 한다. 그러나 MNOS(Metal Nitride Oxide Semiconductor)기술과 비결정 반도체기술에 의해 실현 가능해진 EAROM-(Electically Alterable ROM) 혹은 EEPROM(Electically Erasable ROM)은 회로에서 분리시키지 않고도 전기적으로나 부분적으로 그 내용을 변경시킬 수가 있다. 액세스시간은 15~20 nsec 정도이며, 라이트하는 시간은 2~10 msec까지 단축되었다. 이것은 소프트웨어 제어하에 전기적으로 기억내용을 변화시킬 수 있으므로 RMM(Read Mostly Memory)으로 취급되기도 한다.

그러나 RAM은 기억내용을 리드/라이트하는 시간이나 방법에 큰 차이가 없으며 전원이 짧은 순간이라도 끊어졌을 경우 그 기억내용을 잃게 되는 반면 EAROM(혹은 EEROM)은 리드시간과 라이트시간에 큰 차이가 있고 그 방법도 차이가 나며 전원이 끊어져도 기억된 내용을 상실하지 않는다는 등의 차이가 있다.

자기코어 메모리와 MBM

자성재료를 이용한 기억장치는 기억내용의 비휘발성(자연적으로 기억내용이 상실되지 않는 특성)이며 리드/라이트 동작시만 전력소모가 발생한다는 유리한 점이 있다.

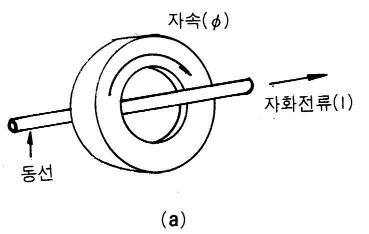
자기코어 메모리는 작은 도너츠형의 자화물질로 그림 5(a)와 같은 모양과 (b)와 같은 자화특성을 갖는다. 이 자기코어는 지름이 0.3㎜~0.7㎜의 마그네슘과 망간 및 자철의 혼합분말을 압축하여 만든 페라이트(Ferrite)이다.

도선에 전류가 흐르면 도선 주위에는 플레밍의 오른손법칙에 의하여 자력선이 발생하고 그림 5(a)와 같이 도선에 자기코어를 끼워 놓았을 경우 자기코어는 ϕ와 같이 자화가 되며 일단 자화된 코어는 전류를 제거하여도 거의 그 상태를 유지한다. 이것은 전류가 흘렀던 방향을 기억하고 있는 것과 마찬가지이므로 기억소자로 이용될 수 있는 것이다. 물론 전류의 방향을 바꾸면 자속의 방향도 바뀐다.

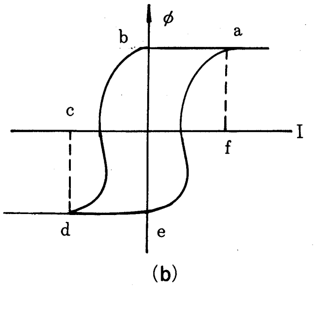
그림 6(a)와 같이 센스(Sense)선을 자화시키고 그 자화의 방향을 바꾸기 위한 구동선과 함께 구성하였을 경우 그림 5(a)에서와 같은 구동전류(자화전류)에 의하여 ϕ와 같은 방향으로 자화된 것을 "1"로 기억되었다고 하면 그 반대방향의 구동전류에 의하여 반대방향의 ϕ로 자화된 것을 "0"의 상태로 기억되었다고 할 수 있다.

이 가정에서 그림 6(a)에서와 같이 구동전류를 흘려 주었을 경우 "1"로

〔그림 5〕 자기코어와 자화특성

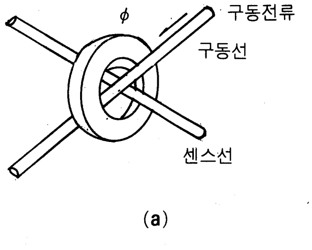


(a)

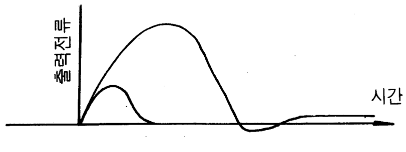


(b)

〔그림 6〕 센스선과 출력전류



(a)



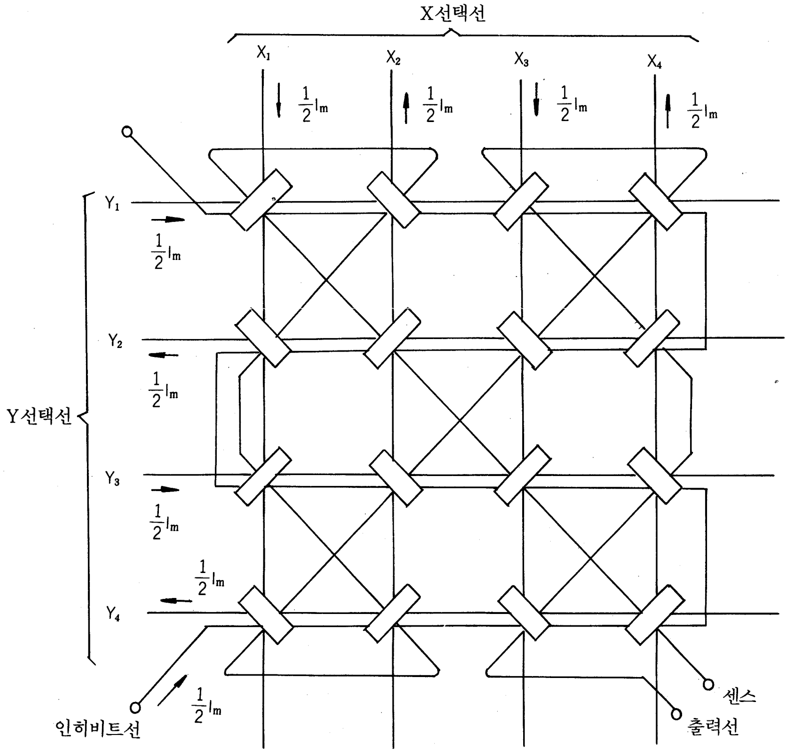
(b)

기록된 상태의 자속 ϕ는 그 방향이 구동전류에 의하여 반전된다. 그러나 "0"의 상태로 기록된 자속 ϕ는 방향전환이 일어나지 않는다. 자속 ϕ가 방향전환을 할 때 자속의 변화가 생기며 이때 센스선에는 기전력의 발생과 함께 순간적인 전류가 흐른다.

즉 "1"로 기록된 상태에서는 "1"로 기록하기 위한 전류와 반대방향의 구동전류에 의하여 센스선에 출력전류를 발생시키나 "0"으로 라이트된 상태에서는 이 방향의 구동전류에 의하여 출력전류가 발생하지 않는다. 구동선의 전류방향을 그림 5(a)에서와 같이 하였을 경우 "1"로 라이트하기 위한 구동선 전류라면 이와 반대방향으로서의 구동선 전류는 구동선 전류라 할 수 있다.

이러한 방법으로 라이트와 리드동작을 하는 자기코어 한개는 1비트의 정보를 기억할 수 있으며 여러 비트의 기억을 하기 위하여는 다수의 코어가 필요하다. 이 다수의 코어 각각을 구분하여 라이트와 리드를 하기 위해서는 그림 7과 같이 X선택선, Y선택선(구동선) 및 센스선 그리고 인히비트(Inhibit) 선을 구성시킨다.

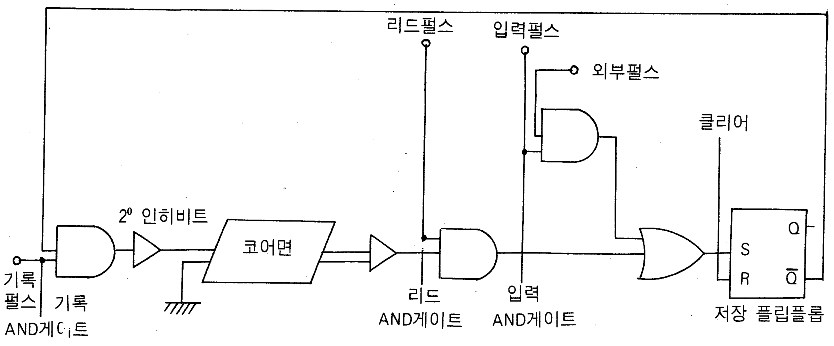
〔그림 7〕 전류 일치방식의 기억장치



여기서 라이트된 정보를 리드할 때 그 기억내용(자화방향)은 변경된다. 즉 "1"로 라이트된 내용은 리드시 "0"이 된다. 이와 같이 리드시 기억내용이 변경되는 것을 파괴형 리드방식이라 하며 이런 자기코어 메모리에 그림 8과 같은 회로를 부가하여 읽을 때 파괴되는 기억내용을 원상 복귀시키도록 한다. 이렇게 하였을 경우 비파괴리드(NDRO; Non Distructive Read Out) 메모리가 된다.(전술한 IC메모리는 모두 NDRO 메모리이다)

이 자기코어 메모리는 사용자들이 편리하게 이용할 수 있도록 기판위에 상품화되어 나오고 있다. 그러나 이것은 IC 메모리에 비하여 동작시간이 길고(0. 1~수msec), 부피에 비하여 기억밀도가 작으며 제작비용이 비싼 단점이 있어 특수 분야에서나 이용되고 있다.

〔그림 8〕 리드와 라이트를 할 수 있는 NDRO 기억장치



MBM(Magnetic Bubble Memory)은 최근 저속컴퓨터에 자기디스크와 테이프 메모리 대신에 많이 사용된다. 이러한 메모리는 오르토페라이트(Orthoferrite) 나 석류석(Garnet)과 같은 물질로 얇은 평판을 이루고 있다. 이 평판형에 자화방향은 평판에 수직이며 평판에 도메인(Domain)이라 불리는 꾸불꾸불한 형태가 그림 9(a)와 같이 자연적으로 생긴다. 여기서 서로 인접한 도메인은 방향이 반대가 되며, 만일 외부에서 바이어스 자장 H_s가 평판의 표면에 수직으로 작용한다면 그림 9(b)와 같이 도메인의 자화방향은 H_s의 방향과 반대가 되며, 크기가 줄어들고 원통 형태가 된다. 이런 유통형의 도메인을 자기버블(Magnetic Bubble)이라고 하며 이의 지름은 보통 1~10μm

정도가 된다.

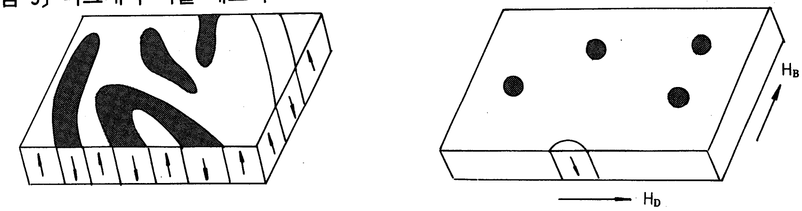
그림 9에 외부에서 구동자장 H_D를 평판표면에 평행으로 작용시키면 버블을 높은 속도로 움직이게 된다. 일반적으로 구동자장은 일정한 속도로 회전하게 되어 회전자장을 형성하며 이러한 회전자장의 발생은 전자석에 의해 발생시킨다. 그리고 평판의 표면에 부드러운 마그네틱 퍼말로이(Permalloy)를 직선적인 트랙으로 놓으면 버블은 이 트랙을 따라 움직이게 된다. 이때 트랙 아래에 있는 버블은 자장에 의하여 일정한 방향으로 계속 움직이게 된다. 그림 10 에서 T와 바형태를 갖춘 퍼말로이 트랙이 서로 번갈아 위치하고 있으며 회전드라이브 자장 H_D는 퍼말로이 요소내의 H_d에 평행되게 자계를 형성하게 된다.

T와 바의 양끝 부분과 T의 중간부분은 H_D에 따라 N극 혹은 S극으로 되며 자석버블은 작은 자석과 같이 되고 두 극중에 어느 한극은(그림에서는 자석 버블의 S극) 퍼말로이 트랙이 놓여 있는 표면에 있게 된다. 따라서 버블은 T나 바의 N극에 끌리게 된다. 그림 10은 H_d의 한 회전에 의하여 버블이 어떻게 이동하는가를 보여준다. 이와 같은 원리를 이용하여 버블 메모리 장치는 버블이 주위가 밀폐된 상태에서 일정한 속도로 회전할 수 있도록 트랙이 형성되어 있다.

이 경우 트랙은 시프트 레지스터 역할을 한다. 그림 10에서 T-바 트랙에서 한쌍의 T와 바는 하나의 기억 셀(Cell)을 형성한다. 각 셀에서의 버블의 유무는 논리 "1"과 "0"상태에 해당하며 데이터를 셀속에 기억시키기 위해서는 버블을 트랙속으로 이끌어 주는 버블 발생장치와 버블을 제거시키는 버블소거장치(Annihilator)가 있어야 한다.

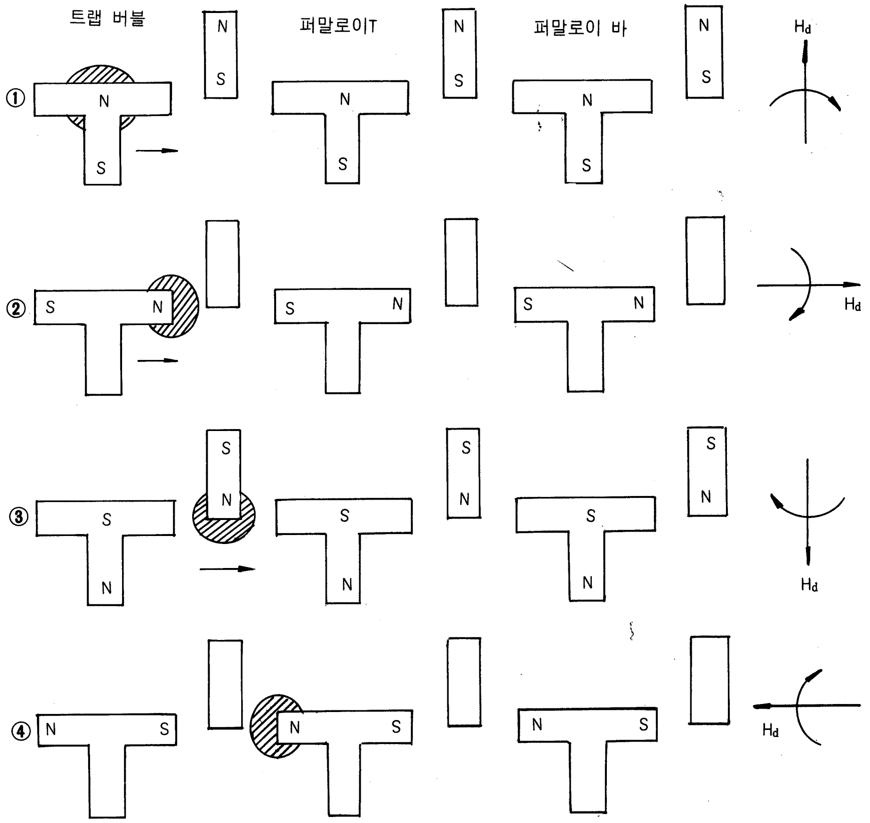
트랙을 매우 빠른 속도로 회전하는 버블을 검출(리드)하는데는 고도의 기술이 필요하다. 특히 버블의 크기가 1μm 정도로 작고 자기 에너지 역시

〔그림 9〕 마그네틱 버블 메모리



(a) 자화방향　　(b) 외부에서 자장이 수직으로 작용할 때

〔그림 10〕 T-바 트랙을 통한 자기버블의 이동



매우 약하기 때문에 이것을 검출하는 것은 매우 어렵다.

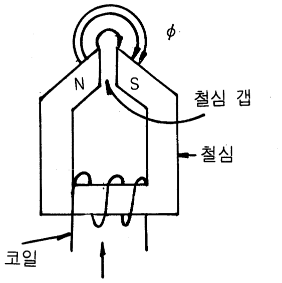
검출방법은 파괴검출 방식과 비파괴 검출 방식이 있다. 비파괴 검출기술로는 전자기적 유도방식, 직접광학검지 방식, 홀(Hall) 효과에 의한 검출방식 등이 있다.

MBM은 최근에 실용화되기 시작했으며 이의 제작에는 집적회로 제작기술이 사용되므로 자기코어 메모리에 비하여 비트당 생산단가가 매우 저렴하다. 한 칩당 기억용량은 1M비트 정도이며 가동부분이 적기 때문에 자기디스크나 테이프 메모리보다 신뢰성이 높아 향후 전망은 좋으나 처리기술이 어려워 인터페이스 등에 난점이 있다.

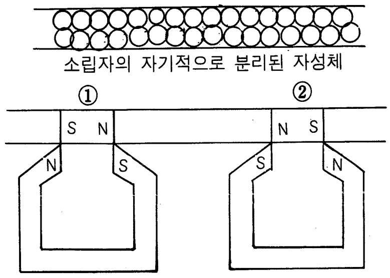
보조기억장치

앞에서 설명한 메모리들은 기계적으로 움직이는 부분이 없으므로 동작속도가 매우 빠르며 부피가 작고 신뢰도

〔그림 11〕 자성체의 자화원리

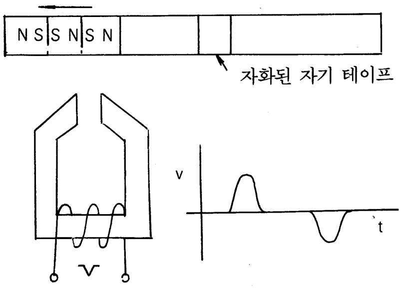


(a) 좁은 갭이 있는 전자석



(b) 전자석의 갭에 의한 자화

〔그림 12〕 자속변화를 이용한 기억장치



가 높다는 장점이 있다. 그러나 메모리 용량당 단가나 메모리내용의 이동, 복사, 대용량화 등에 문제가 있다.

이러한 문제점들을 해결할 수 있는 기억장치들로 자기테이프 메모리, 자기디스크 메모리, 자기드럼 메모리, 광디스크 메모리 등이 있다. 이들은 빠른 스위칭 동작을 중요 특성으로 하는 컴퓨터나 디지탈 시스템에서 직접적으로 적합한 메모리를 사용하기에는 부족하지만 보조메모리장치로서 훌륭한 일익을 담당할 수 있는 기억장치들이다. 이들은 때로는 컴퓨터 등에서 고속 입출력 장치의 역할을 하는 경우도 있다.

각각 이들의 구조와 동작 및 특징들을 살펴보기로 한다. 먼저 이들 대부분이 자성체 자화특성을 이용하므로 디지탈신호(혹은 정보)를 라이트하거나 리드하는 방식에 관하여 알아 보자.

●디지탈 라이트방식

코일에 전류를 일정한 방향으로 흘려줄 경우 코일주변에는 일정한 방향의 자장이 발생한다. 이 때 코일속에 철심코어를 삽입하면 코일주변의 자속을 철심에 모을 수 있고, 철심은 전자석이 된다. 코일에 흐르는 전류의 방향을 바꾸어 주면 철심의 자장방향도 바뀌게 된다.

철심 끝에 작은 사이를 두어 철심갭을 그림 11(a)와 같이 하였을 경우 철심 갭사이에는 자장이 ϕ처럼 분포하게 되며, 이 철심 갭에 인접하여 보자력이 크고 자기적으로 분리된 소립자의 분말 자성체를 그림 11(b)와 같이 놓았을 때 그 자성체는 그림 11과 같이 국부적으로 자화된다. 이 철심에서는 자성체를 멀리하여도 일단 자화된 상태는 보존된다. 다시 전자석의 위치를 ①에서 ②로 옮겨 전류방향을 바꾸어 주면 갭에서의 자극의 방향이 바뀐 상태로 ②의 위치가 자화된다.

이와같이 자화된 상태는 전자석의 코일에 흘려주는 전류의 방향에 따라 NS 혹은 SN으로 자화되어 보존되므로 이 자성체는 전류의 방향을 기억하고 있는 것이 된다. 자성체를 이동시키며 전자석의 코일전류방향을 바꾸어 주므로서 많은 전류방향 정보를 라이트할 수가 있다. 여기서 자성체는 플라스틱 리본(Plastic Ribbon), 플라스틱 디스크(Plastic Disk), 알미늄 디스크(Aluminum Disk) 등에 자성체인 산화철을 발라서 만든다(그림 12 참조).

한편 앞에서 예를 든 전자석의 코일 양단을 개방하고 철심의 갭에 자속의 변화를 주면 코일에는 기전력이 발생한다. 즉 철심의 갭 근처에 영구자석을 근접시켰다 멀리하면, 자속은 가까울 때 커졌다가 멀어질 때 작아지는 자속의 변화가 생기며, 이로써 코일에는 기전력이 발생하는 것이다. 그리고 자화된 방향에 따라 기전력의 방향이 달라지고 이로서 라이트할 때 상태의 전류방향을 리드해낼 수가 있다. 코일에 철심을 삽입하고 철심이 좁은 갭을 갖고 있어 앞에서 설명한 바와 같이 자성체에 전류 정보를 자기 정보로 라이트하고 자기정보를 전류정보로 변환하는 장치를 각각 라이트 및 리드헤드(Head)라고 한다.

이러한 원리를 이용한 기억장치는 자성체, 즉 자기테이프 또는 자기디스크를 가동시키고, 헤드는 고정시켜 동작하도록 하는 경우와 자성체는 고정시키고 헤드를 가동시켜 동작하도록 하는 경우를 생각할 수 있는데 양자 모두 기계적인 가동부분이 필요하다. 이로 인하여 동작시간은 기계적인 동작을 필요로 하지 않는 IC메모리 등에 비해 상당히 늦다.

●자기테이프 메모리

우리 가정에서 사용하는 녹음테이프나 카세트 테이프와 같은 원리와 형태로서 표준형은 폭이 0.5인치이고 한권의 길이는 2400ft이다. 일반 음성용 녹음에서는 단일 채널(스테레오 인 경우는 2채널)로 헤드가 녹음, 재생용으로 각각 한개(스테레오의 경우는 2쌍)가 있으나 디지탈용에는 보통 7쌍의 헤드 7채널형으로 되어 있다.

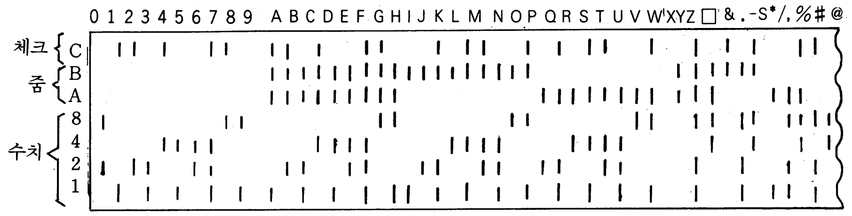
즉 테이프에 행을 7채널로(또는 트랙이라고도 함) 구분하고 열을 프레임으로 구별한다. 자기테이프는 기록하는 속도에 따라 200~1600CPI(Characters Per Inch)의 범위에 있으며 2400ft짜리 테이프 하나는 20M바이트의 정보를 수록할 수 있다. 기록정보의 전달속도는 15,000~350,000CPS(Characters Per Second)이며, 테이프는 2~5만번까지 지워서 다시 사용할 수 있다. 7채

널 테이프의 예를 그림 13에 보였다.

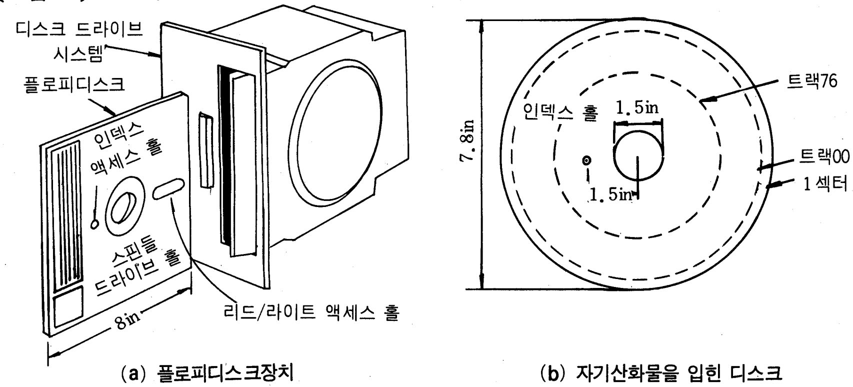
테이프는 순차적으로 정보를 라이트함으로 이를 리드할 때 효과적으로 기록된 정보를 찾을 수 있도록 하기 위하여 레코드(Record)라는 정보군을 만들고 레코드와 레코드 사이에는 간격(Interrecord Gab)을 두어 구별하도록 한다. 한 레코드는 대개 일정한 형태로 포맷(Format)되어 있으며, 이 포맷에는 필요한 정보외에 레코드번호 착오검출용 데이터 등이 포함된다.

마그네틱 테이프 메모리는 많은 정보를 한꺼번에 라이트할 수 있으며 테이프의 보관, 관리, 이동이 간편한 장점이 있으나 순차기록, 리드방식으로 동작되므로 테이프의 어느 한쪽 끝단의 정보를 처리하다가 또 다른 쪽 끝단의 정보를 처리하고자 할 때 중간부분의 불필요한 정보의 통과시간이 소요되므로 다음에 설명하고자 하는 디스크형에 비해 처리속도가 늦다.

〔그림 13〕 7채널 자기테이프



〔그림 14〕 플로피디스크



(a) 플로피디스크장치

(b) 자기산화물을 입힌 디스크

●자기디스크 메모리

디스크 메모리는 전축의 음반처럼 생긴 원반의 자성재료를 칠하여 이 원반의 디스크를 고속으로 회전하도록 하며, 정보는 동심원의 트랙(라이트, 리드용 헤드밑을 통과하는 궤도)상에 저장된다. 각 트랙은 섹터(Sector)로 구분되며, 한 디스크내의 정보를 리드하거나 라이트할 경우 트랙번호와 섹터번호에 의해 이루어진다.

이러한 디스크 기억장치는 헤드와 디스크의 형식에 따라

① 헤드 고정, 디스크 고정방식
② 헤드 이동, 디스크 고정방식
③ 헤드 이동, 디스크 이동방식

으로 분류할 수 있으며 고정헤드방식은 다중 헤드방식으로 트랙마다 헤드를 설치하는 것이며, 헤드 이동방식은 디스크 한면에 한개의 헤드를 설치하여 트랙을 따라 헤드를 이동시킨다.

디스크는 교환을 할 수 없는 고정식과 디스크를 수시로 교환할 수 있는 방식이 있으며, 한장의 디스크로 되어 있는 것과 여러장으로 구성되어 있는 것이 있다. 여러장으로된 교환 디스크를 디스크 팩이라고 한다.

알미늄과 같은 견고한 원반면에 자성체를 칠하여 만든 하드디스크와 플라스틱 재질(사진 필름 재질과 같이 부드러운)의 원반에 자성재료를 칠한 유연성의 플로피 디스크가 있다.

대형 메모리용량을 요하는 곳에서는 HDD와 같은 고정방식을 쓰며 소용량에서는 플로디스크를 사용한다. 즉 하드형일 때는 기록밀도를 높일 수 있다. 그러나 유연성, 이동성방식에 비하여 메모리 비트당, 단가는 비싸지고 취급이 간편하지 않다. 퍼스널 컴퓨터 등에서 많이 이용되는 플로피디스크에는 8", 5¼", 3. 5" 등 다양한 형의 플로피디스크가 이용되고 있으며 디스크당 기억용량은 수100K 바이트에서 수M 바이트 정도이다. 그림 14는 플로피디스크의 구동장치 및 디스크이다.

예로서 8" 디스켓은 보통 77트랙을 갖고 있으며, 각각의 트랙은 26개의 섹터로 구분되고, 한 섹터는 128바이트의 데이터를 수록하도록 한다. 따라서 8"의 플로피디스크에 총 기억용량은

77×26×128=256,000(바이트)

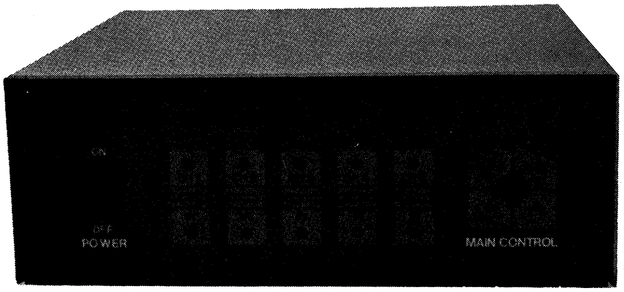
가 된다.

이 외에 자기 드럼메모리 등도 있으나 현재는 많이 이용되지 않고 있으며 기록매체의 형태가 드럼(Drum)으로 되어 있다는 것외에는 MT나 디스크와 같다고 할 수 있다.

PRINTER 주변기기 전문 메이커

- 컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- 컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- 프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- 컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- 학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!



※ K-P-345A는 컴퓨터 프로그램 보호법에 의해 등록된 제품입니다.

K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3 : 1~5 : 1
출력 PRINTER	1대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1 : 1 or 2 : 1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



● DATA SWITCH
- 전자식 완전자동 (2 : 1~8 : 1 or 64 : 1)
- 전자식 반자동 (2 : 1~8 : 1, 4 : 2)
- Serial 완전자동(4 : 1, 8 : 1)

● DATA BUFFER
- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

● DATA CONVERTOR
- Serial→PARALLEL
- PARALLEL→Serial

● 각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울·종로구 종로3가 175-4 (현대APT1309호)
전 화 : 277-1739

● 상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다.●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천 : 바이너 리컴퓨터 (032) 762-5544
대구 : 범진 컴퓨터 (053) 252-2211
마산 : 한길 컴퓨터 (0551) 94-5555
원주 : 중부 시스템 (0371) 43-7979
(기타 지역 전문 취급점 모집중)

제어용 마이크로컴퓨터를 이용한 사인보드

이우섭* · 전하경** / *인하대학교 전자공학과 3년, **인하대학교 자동화공학과 3년

필자가 몸을 담고 있는 "인하대학교 컴퓨터클럽"에서는 하드웨어에 관심있는 신입생을 대상으로 교육을 실시하고 있다. 따라서 컴퓨터를 이해시키고, 자동제어기술을 익히며, 기계어와 어셈블리프로그램을 이해하고 작성하는 것을 교육의 목표로 삼고, 하드웨어 교육의 결정체라 할 수 있는 마이크로컴퓨터를 중심으로 교육을 진행하고 있다.

이러한 연유로 필자도 써클에서 직접 마이크로컴퓨터를 제작하는 경험을 쌓았고 이를 토대로 지금의 사인보드를 제작하였다.

사인보드(Sign Board)란 우리가 원하는 문장이나 기호등을 어떤 화면을 통해서 사람들에게 보여 주는 것으로 우리 주변에서 쉽게 찾아 볼 수 있다.

예를 들어 지하도나 지하철역같은 곳에 설치되어 있는 사인보드는 시민들에게 경각심을 일깨우는 표어를 내보내고 물고기가 한글자씩 잡아 먹는 등의 재미있는 모양을 하고 있고, 서울시내 일부 큰 서점안에 설치되어 있는 사인보드는 시민들에게 여러가지 정보를 알려 주고 있다.

이러한 사인보드를 구동하는 방법은 다양하나 여기서는 제어용 마이크로컴퓨터를 이용하여 구동하는 방법을 사용하였다. 그리고 여러명이 몇개의 모듈로 나눠 제작하였기 때문에 제작에 참여한 사람들에게는 컴퓨터에 자신을 가질 수 있었고, 사인보드를 만들어 봄으로서 주위에서 흔히 볼 수 있는 사인보드에 친근감을 가질 수 있는 기회가 되었다.

컴퓨터에 관심은 있으나 엄두가 나지 않는 사람이 있다면 이번 기회에 컴퓨터와 인터페이싱 기술을 익힐 수 있는 기회가 되길 바란다.

제작에 들어가서

사인보드의 전체구성도는 **그림 1**과 같이 크게 나누어보면 마이크로컴퓨터, 카운터(Counter), 디지탈 시계, 센서, LED 드라이버로 나뉘어져 원하는 메시지(Message)를 내보내고, 시간이나 그 때까지의 방문자 수를 출력하도록 구성되어 있다.

그러나 이 사인보드는 본 써클을 포함한 23개교의 컴퓨터클럽이 연합하여 활동하고 있는 UNICOSA(전국 대학 컴퓨터써클 연합회)가 주최하는 전시회에 출품했던 작품이기 때문에 UNICOSA 전시회에 관련된 데이터만 출력되도록 프로그램을 작성하였으므로, 직접 제작해보고 싶은 독자는 뒤에 설명하는데로 몇가지만 바꾸면 된다. 그림 1을 보면 8행 48열에 LED디스플레이(Display)가 화면을 이루고 있다. 이는 메인보드내의 ROM 2764에 내장되어 있는 프로그램에 의해 구동된다. 프로그램은 먼저 기록되어 있는 데이터를 내보낸뒤 시각 및 방문객의 총인원수를 내보낼 수 있도록 작성되어 있다.

메인보드와 디지탈시계, 카운터, LED 드라이버와의 인터페이싱에는 Z80 PIO 3개를 이용하였다.

여기에는 +5V전압을 필요로 하여 퍼스널 컴퓨터용으로 시중에 판매되고 있는 6A, 5V의 스위칭 레귤레이터(Switching Regulator)를 사용하였다.

스위칭 레귤레이터의 구입이 힘든 사람은 **그림 2**의 회로를 통해 자작할 수 있다. 7805(레귤레이터) 하나만으로는 전류가 부족하므로 2N2955를 버퍼(Buffer)로 사용하여 충분한 전류를 사용할 수 있도록 하였다.

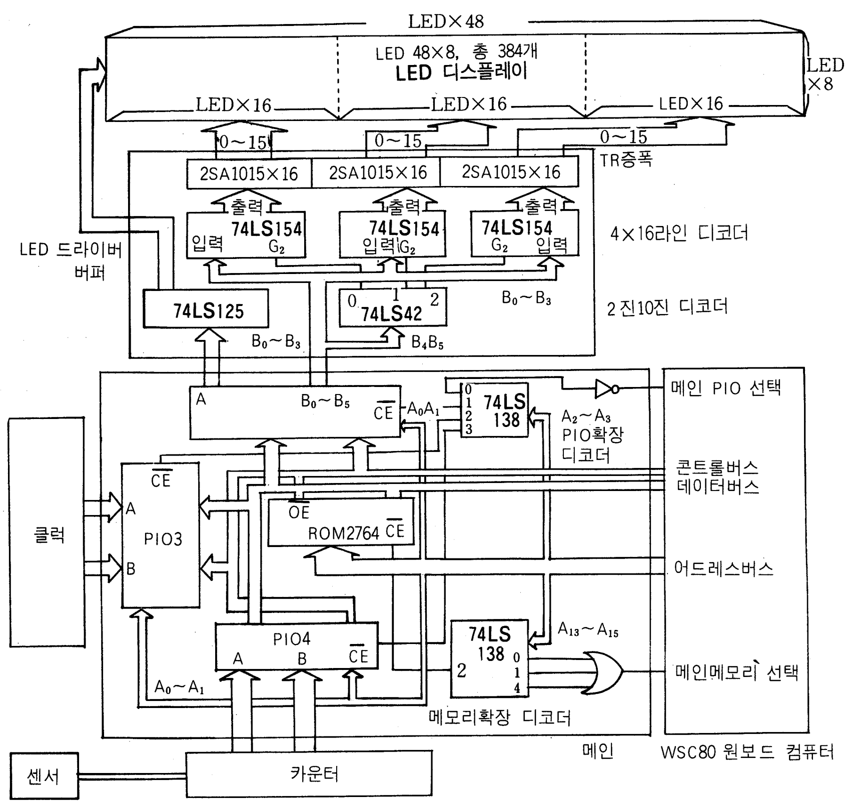
그럼 각 모듈별 제작과정을 살펴보자.

마이크로컴퓨터

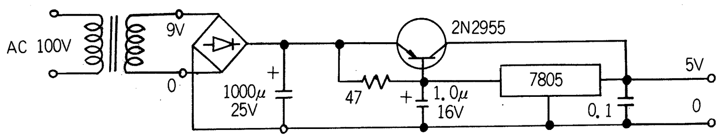
독자들이 가장 원하는 것이 원보드 컴퓨터의 정복이라 생각된다. 컴퓨터 하면 특히 마이크로컴퓨터의 경우 CPU의 동작과 메모리확장법 및 I/O인터페이싱이 그 주를 이룬다고 보아도 무리는 아니다. 우선 CPU를 이해하도록 하자. 여기서는 Z80 CPU를 이용하였으므로 Z80 CPU만 설명하겠다. Z80 CPU의 내부 계통도는 **그림 3**과 같다.

여기서 ALU란 Arithmetic Logical Unit의 약자로 '산술연산 논리기구'라고 번역할 수 있다. 이는 CPU내에서는 가산(Add), 감산(Subtract), 논리곱(AND), 논리합(OR), 배타적 논리합(Exclusive OR), 비교(Compare), 좌우시프트 및 회전(Shift and Rotate), 인크리먼트(Increment), 디크리먼트(Decrement), 비트 세트(Set Bit), 비트 리세트(Reset Bit), 비트 테스트(Test Bit)로 연산의 종류를 세분한다. Z80 CPU의 핀배열도와 핀설명은 **그림 4**와 같다. 메모리 및 I/O인터페이싱에

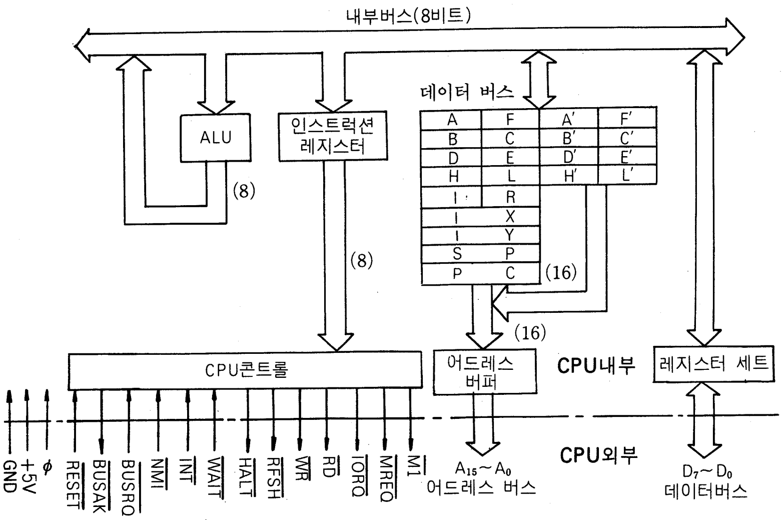
〔그림 1〕 사인보드의 전체구성도



〔그림 2〕 파워 서플라이



〔그림 3〕 Z80 CPU의 내부구성도



대해서는 메인보드부분에서 설명한다.

마이크로컴퓨터를 공부하는 사람들은 직접 제작하거나 키트를 이용할 수도 있는데 각기 일장일단이 있다. 자작의 경우는 필요한 부분만을 제작하므로 특별한 목적에 유용하고 제작비가 적게 든다는 장점이 있으며 키트의 경우는 광범위하게 이용할 수 있으나 필요하지 않은 부분이 많이 생기는 단점이 있다.

필자의 경우 자작품이 있지만 이는 부피가 너무 커서 전시회용으로 제작된 본 사인보드에는 부피를 줄이기 위해 시중에서 구하기 쉬운 WSC-80A를 구입하여 사용하였다.

컴퓨터는 CPU와 RAM만을 이용하고 있으므로 CPU, RAM, CPU클럭부만을 만들어 컴퓨터대용으로 쓸 수도 있으며 다른 컴퓨터라도 Z80 CPU와 RAM만 8000H 번지부터 쓸 수 있다면 사용해도 무방하다. 여러분의 컴퓨터구조에 대한 이해를 돕기 위해 WSC-80A의 회로도를 **그림 5**에 싣는다.

메인

메인은 컴퓨터의 확장이며 LED드라이버, 디지탈시계, 그리고 카운터와의 인터페이싱을 하는데 목적이 있으며 메모리 확장에도 그 의의가 있다.

이 부분에서 우선적으로 알아야 될 것은 PIO와 PIO확장법 및 메모리 확장법이다.

Z-80 PIO(Parallel I / O)는 Z-80 패밀리중 가장 먼저 발표된 것이다. 8255A (PPI)는 세개의 포트를 갖는 반면, Z-80 PIO는 두개의 포트를 가지며, 4개의 입·출력모드, 쌍방향모드, 비트모드를 갖고 있다. 모든 입·출력라인은 TTL콤패티블이다. 우선 PIO의 핀배열과 내부 구성도를 **그림 6**에 보였다.

그러면 PIO의 동작에 대해서 알아보자. 우선 CE(Chip Enable)는 이 단자가 "0"일 때 동작상태이고, "1"일

때 부동작 상태이며, $B/\overline{A}$는 이 단자가 "0"이면 A포트이고, "1"이면 B포트를 선택한다. 그리고, $C/\overline{D}$는 이 단자가 "1"이면 제어어 레지스터가 선택되어 "0"으로 하면 데이터 입·출력 레지스터가 선택된다. $B/\overline{A}$에 A_1, $C/\overline{D}$에 A_0를 연결시켰을 때 각 포트의 번지지정과 모드지정은 **그림 7**과 같이 한다.

제어어 레지스터에 제어어(Control-word)를 써넣어서 그림 7(b)와 같이 모드지정을 한다. 여기서는 모드 0, 모드 1에 대해서 주로 알아 보자.

예로 A포트를 모드 0(전바이트 출력) B포트를 모드 1(전바이트 입력)으로 지정하고 싶다면 A의 제어어 레지스터에 제어어 '0F'를 써 놓고 B의 제어어 레지스터에 제어어 '4F'를 써 넣으면 된다.

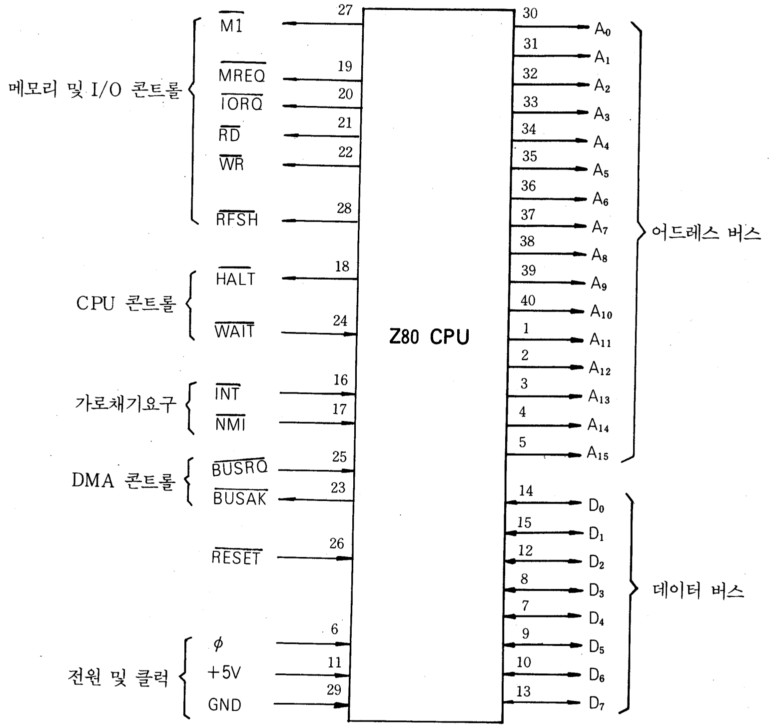
그림 8의 회로와 프로그램은 B포트로 입력을 받아 A포트로 내보내 B포트의 스위치로 A포트의 LED를 점등시키는 회로이다. 여기서 OUT은 CPU에 있는 데이터를 PIO로, IN은 PIO에서 CPU로 이동시키는 명령어들이다.

이번에는 PIO확장에 따르는 모드 지정방식을 **그림 9**를 통해 알아 보자. PIO확장은 A_0~A_1은 동일하게 모든 PIO에 연결시키고 A_2~A_7까지의 값으로 PIO를 선택한다. A_2~A_3의 출력을 2진 10진 디코더의 입력에 연결하고 출력을 각 PIO의 $\overline{CE}$에 연결시키면 된다. 00~03은 PIO1을, 04~07은 PIO2을, 08~0B는 PIO3을 선택한다.

그림 9의 회로에서는 PIO 1의 A포트에서 데이터를 받아 PIO 2의 A포트의 LED를 점등시키고, PIO 2의 B포트에서 데이터를 받아 PIO 1의 B포트의 LED를 점등시키는 회로이다. 여기서 PIO 1의 A포트, PIO 2의 B포트는 전입력 모드이고, PIO 1의 B포트, PIO 2의 A포트는 전출력 모드이다.

사인보드에서는 모드 2, 모드 3(쌍방향, 비트모드)와 핸드세이크, 인터럽트는 사용하지 않으므로 설명하지 않겠다. 메모리도 PIO와 같은 방법으로 확장시킨다. 2764(8K ROM)을 쓰는 경우, **그림 10**과 같이 확장시킨다.

〔그림 4〕 기능별로 나눈 Z80의 신호



그러면 메인부분에 대해서 알아보기로 하자. 메인은 회로도를 보면 알 수 있듯이 디지탈클럭, 카운터, LED드라이버, 마이크로컴퓨터의 연결에 필요한 부분이다. 이를 연결하기 위해 ROM을 하나 확장시킨 모듈이다. **그림 11**에 메인 부분의 회로도를 보였다.

A_2 A_3을 이용해 PIO를 확장하며 A_{13}~A_{15}로 메모리를 확장한다. PIO 2의 A포트는 LED 디스플레이의 8개의 행에 이용되고 B포트는 다시 디코더를 통해서 48열의 선택에 이용된다. PIO 3의 A포트는 디지탈클럭의 "분", B포트는 "시"의 입력에 이용되며, PIO 4의 A포트는 카운터의 아래 두자리, B포트는 위 두자리 입력에 이용된다.

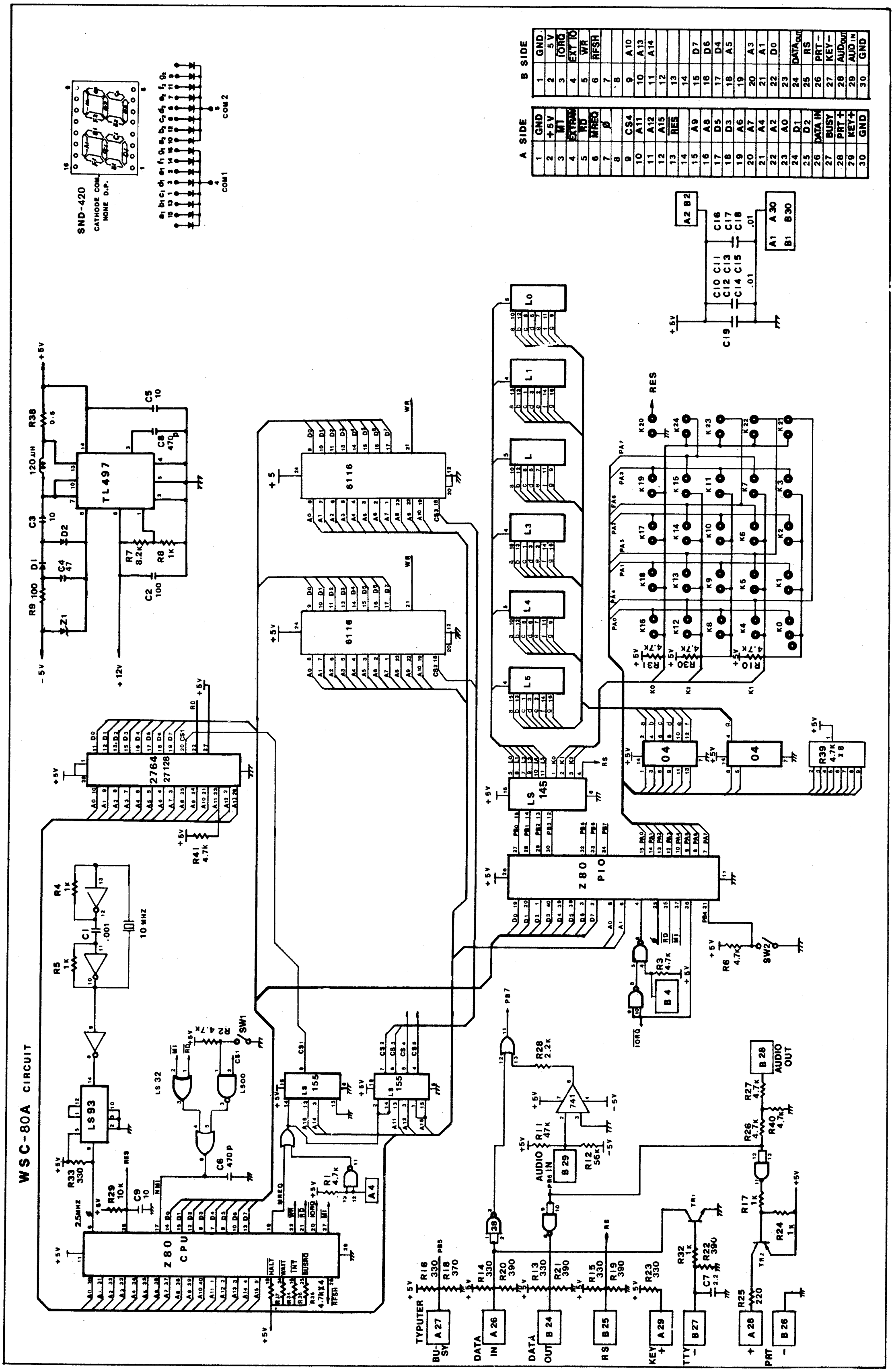
LED 드라이버

LED 드라이버는 8행 48열의 LED 디스플레이를 제어하기 위해 제작한 것이다. 우선 LED와 드라이브방식 및 디코더에 대해 알아 보고 회로를 보기로 하자.

LED(Light Emitting Diodes)는 전류가 흐를 때 빛을 내는 다이오드로 발광 다이오드라고 한다. 기호는 **그림 12**와 같고 극성이 있어 이에 유의하여야 한다. 극성은 긴 쪽이 애노드(Anode) 즉 ⊕단자이다.

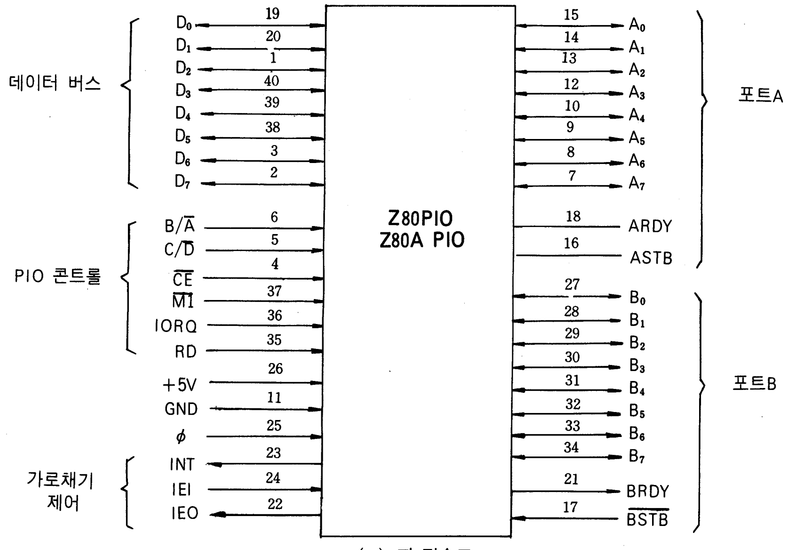
만약 리드선이 잘라져 극을 알아볼 수 없을 경우에는 테스터로 알아 볼 수 있는데, R×10레인지에서 테스트한다. R×1레인지에서는 테스터 자체내에 140mA가 흐르므로 사용하지 않는 것이 좋다. LED는 통상 2V정도의 전압과 10mA~20mA 정도의 전류를 흘려서 발광시켜 주기 때문에 **그림 13**과 같이 보호저항을 달아 전압·전류를 제어해 준다.

7세그먼트 LED는 숫자를 눈으로 볼 수 있도록 LED를 숫자의 형태로 배열하고, 입력된 데이터(신호)에 맞는 숫

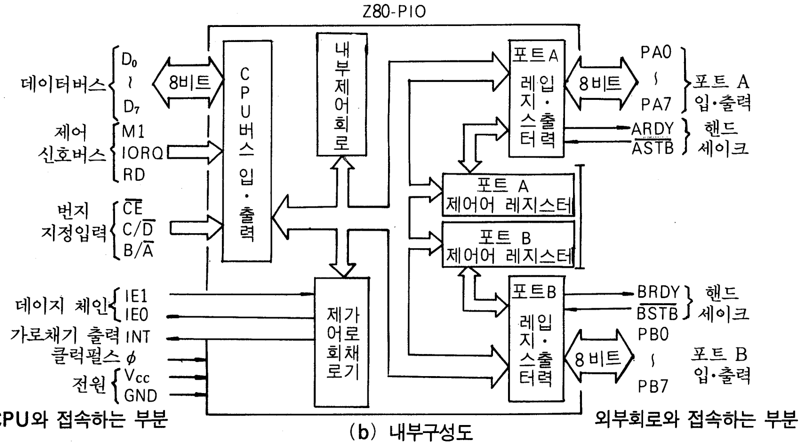


〔그림 5〕 WSC-80A 회로도

〔그림 6〕 Z80 PIO



(a) 핀 접속도



(b) 내부구성도

자로 점등된다. 이 7세그먼트 LED에는 그림 14에 보인 것과 같이 애노드 코먼(Anode Common)형과 캐소드 코먼(Cathode Common)형이 있다.

애노드 코먼형을 써서 7세그먼트 LED의 동작상태를 한번 보자. 그림 15에서 SN7447의 입력단자에 0000H이라는 신호를 입력한면 LED에서는 G단자와 D_p단자만이 소등되고 모든 단자가 점등된다. 그래서 '0'이라는 표시를 나타내게 된다. SN7447에 대한 자세한 사항은 디코더부분을 참고하기 바란다.

2진 코드나 BCD와 같은 코드화된 신호를 10진수나 7세그먼트 신호와 같은, 다른 코드로 변화시켜 주는 것을 디코더라 한다.

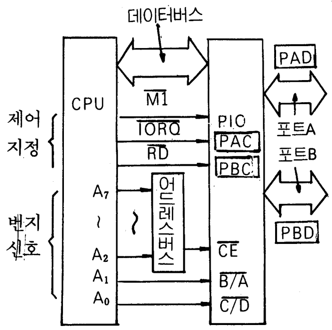
그림 16은 SN7442의 내부를 그린 것이다. 입력에 신호(BCD)가 가해지면 내부의 NAND가 동작하여 입력에 따른 출력핀은 "L"로 된다.

SN7445의 출력회로가 콜렉터인 경우는 출력이 "L"일 때 전류를 많이 흘려보낼 수 있으므로 LED를 직접 연결하지 않아도 무방하다.

이 같이 LED 등의 부하를 직접 연결하여 자유롭게 콘트롤할 수 있는 것을 드라이버라 한 다.

7세그먼트 LED를 2진 코드와 BCD

〔그림 7〕 PIO의 번지지정과 모드지정



① PIO의 포트 A, 포트 B의 번지

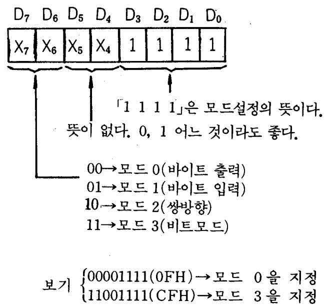
A_1	A_0	어드레스*	PIO내의 지정장소	기 호
0	0	00H	포트 A 데이터 입출력	PAD
0	1	01H	포트 A 제어어 레지스터	PAC
1	0	02H	포트 B 데이터 입출력	PBD
1	1	03H	포트 B 제어어 레지스터	PBC

*A_2~A_7

② 어드레스 지정의 보기(00H~FFH의 범위)

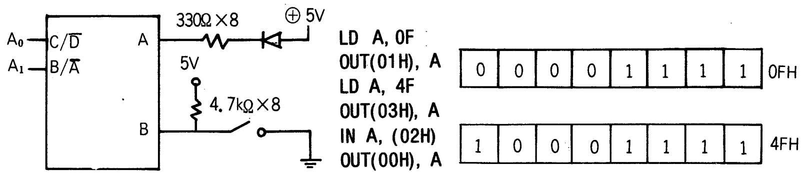
$\overline{CE}$=0이 되는 어드레스								어드	지정한
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	레스	곳
1	1	0	0	1	1	0	0	CCH	PAD
0	0	1	0	0	1	0	1	25H	PAC
1	0	0	0	1	1	1	1	8FH	PBC

(a) PIO 각부의 번지지정

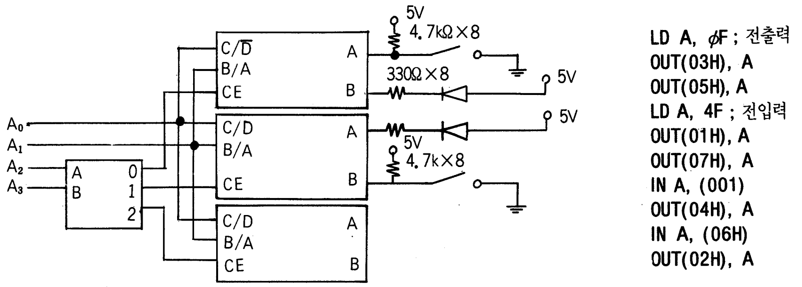


(b) 모드를 지정하는 방법

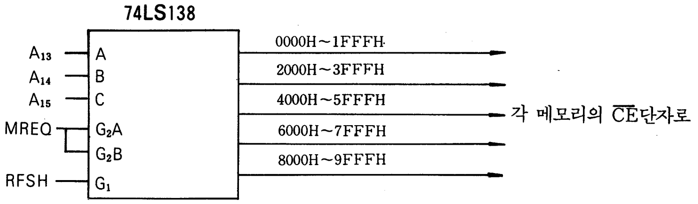
〔그림 8〕 프로그램1



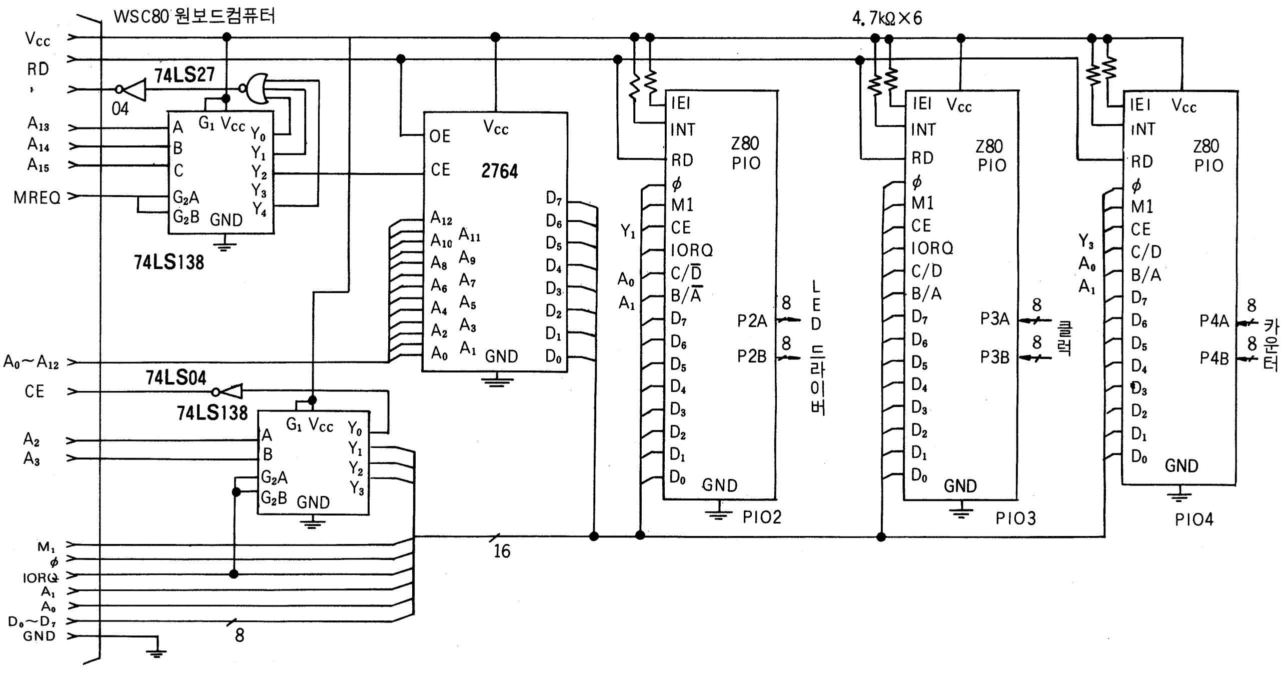
〔그림 9〕 프로그램2



〔그림 10〕 메모리확장



〔그림 11〕 메인의 회로도



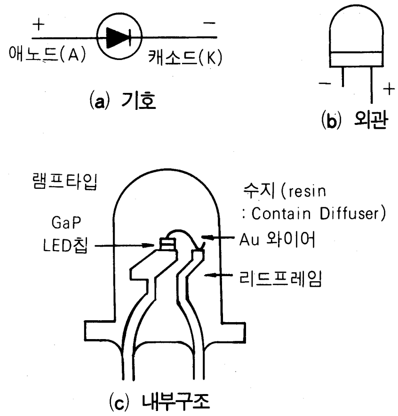
에 따라 동작시키는 데는 디코더가 필요하다. 여기에는 TTL의 BCD 7세그먼트 LED용 디코더 / 드라이버 SN7447을 사용하게 된다. SN7447의 핀접속도는 그림 17과 같다. 여기서 3번의 LT는 램프 테스트(Lamp Test)로 보통은 1로 해 놓는다. 이것을 0으로 하면 출력은 모두 ON이 되어 7세그먼트가 점등된다. RBI / RBO도 보통 1로 놓는다. 이것을 0으로 하면 램프 테스트의 경우와 반대로 모두 OFF가 된다.

그림 18에는 LED의 드라이브법을 보였다. 여기에서 LED의 보호저항을 삽입시켜 직접 LED와 연결한 것을 볼 수 있다. 그림 15에서 설명한 드라이브법은 표시기의 하나하나에 신호선을 연결하여 각각에 별도로 신호를 보내고 있다.

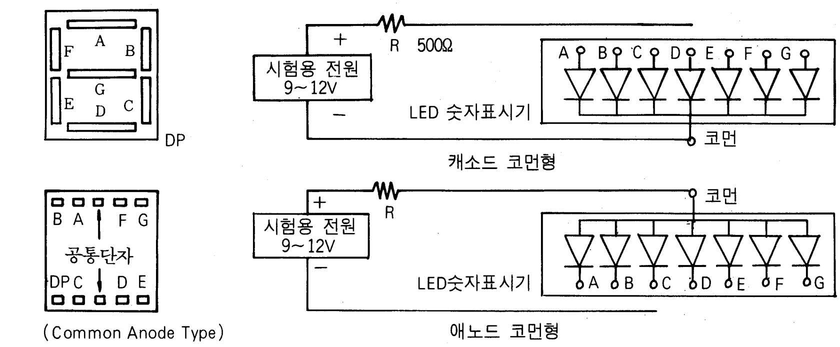
이와 같은 드라이브법을 스태틱 드라이브(Static Drive)법이라고 한다. 이 방법은 알기 쉽고, 드라이브도 간단하므로 사용은 용이하나 표시기의 수가 많아지면 신호선의 수도 많아진다.

다이나믹 드라이브(Dynamic Drive)

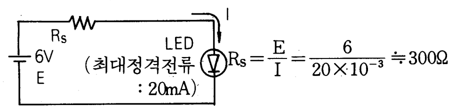
〔그림 12〕 LED의 구조



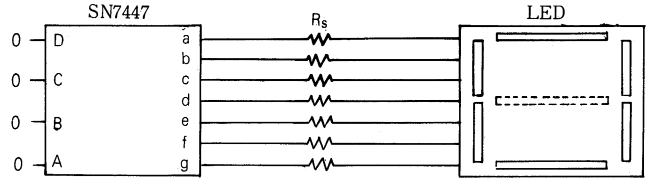
〔그림 14〕 7세그먼트 LED



〔그림 13〕 LED의 전압·전류제어



〔그림 15〕 LED의 점등방법



법의 경우는 표시기가 동시에 점등하는 것이 아니고 자리 드라이브선으로부터 오는 자리 펄스에 의해서 순차적으로 점등해 간다. 즉, 전 표시기에 신호선에서 신호가 보내지지만 점등하는 표시기는 자리 드라이브선으로부터 자리 펄스가 들어 오는 것 뿐이다. 따라서, 표시기의 수에 비해서 신호선의 수가 적게 된다.

이번 제작에서는 7세그먼트 LED에 스태틱 드라이브, LED 디스플레이에는 다이나믹 드라이브방식을 사용했다.

그림 19의 회로도에서와 같이 BCD 카운터가 00H에서부터 09H까지 카운트 업(Count-up)될 때 디코더의 출력은 0~9까지의 반전출력이 나타난다. 그러므로, 디코더의 출력은 언제든지 0~9중의 하나만 "L"이 된다. 즉, 어느 한 순간에 LED들은 하나만 점등된다. 클럭주파수가 높아지면 0에서부터 9까지 순차적으로 LED점등이 반복되므로 적당한 클럭주파수만 만족되면 모든 LED가 점등된 것으로 착시현상이 생긴다.

이해가 되지 않는 독자는 그림 20에서 스태틱 드라이브법과 다이나믹 드라이브법의 차이를 찾아보기 바란다.

그림 21의 LED드라이브 회로도를 보자. 이는 다이나믹 드라이브방식이며, 자리 드라이브를 위해 4진, 16진 디코더 3개를 써서 48열을 만들 수 있었던 것이다. 만약 2진 10진 디코더를 썼다면 디코더 5개를 써서 50열을 만들었을 것이며 아울러 제어 프로그램을 작성하는 데도 더 힘이 들었을 것이다. SN74154가 잘 어울리는 TTL이다. 그림 21은 다이나믹 드라이브 구동방식으로 PIO 2의 B포트 데이터가 00~30H까지 변하면서 번지를 지정하며 만나는 점(열이 선택된 위치에서 A포트의 데이터가 0일 때)만 켜지는 매트릭스 점등방식이다.

TR증폭으로 열의 5V입력을 주고 보호저항을 달지 않는 것은 SN74125의 로우출력 전류가 LED의 정격전류 정도 밖에 안되므로 보호저항을 달 필요는 없다. LED는 모두 8행 48열로 총 384개의 LED를 사용하였다.

이의 동작을 위해서 프로그램에서는 B레지스터를 48열의 열선택에 쓰며 B레지스터는 00H부터 30H까지 카운트업되면서 PIO 2의 B포트로 출력한다. 48개의 열이므로 6개의 비트면 선택이 가능하다. 이는 PIO 2 B_4비트와 PIO 2 B_5비트를 10진수로 디코드시켜 각 SN74154의 G_2에 입력하여 3개의 디코더 중 하나를 선택한다.

PIO 2 B_0부터 PIO 2 B_3비트까지는 각 디코더의 입력선에 동일하게 연결시킨다. 다시 말해서 이 회로는 6~48라인 디코더를 꾸몄다고 보면 된다.

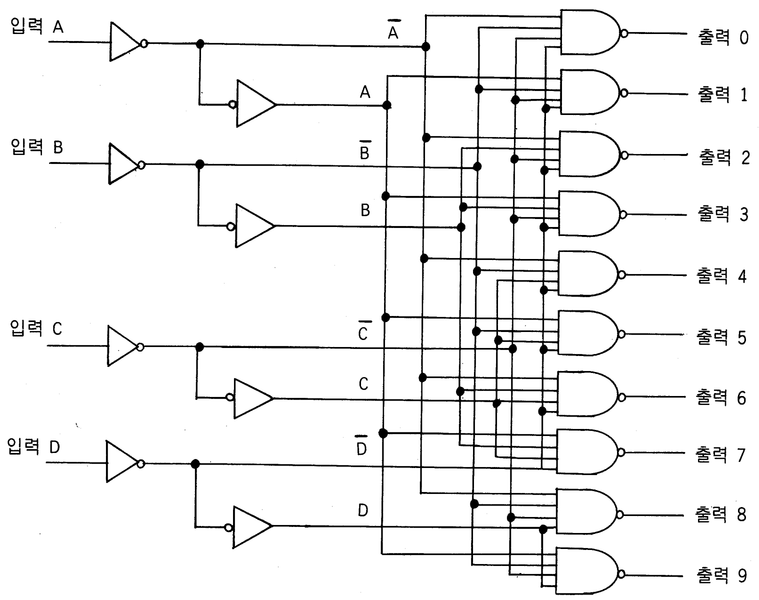
A포트로는 항상 메모리에서 데이터를 읽어서 반전시켜 출력시킨다. 데이터를 입력시킬 때는 켜지는 LED를 1로 보고 입력시켰는데, 회로에서는 데이터가 0일 때만 LED가 점등되므로 반전시켜 주었다.

디지탈시계

시계의 회로를 보기 전에 먼저 몇가지 알아보자. 우선 7세그먼트 LED와 디코더는 앞에서 다루었고 여기서는 카운터에 대해서 알아 보자.

카운터는 입력펄스의 수를 계수하는

〔그림 16〕 SN7442

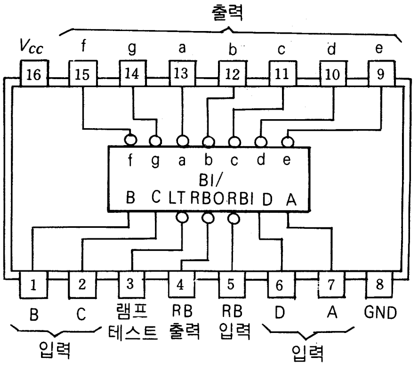


(a) 블록다이어그램

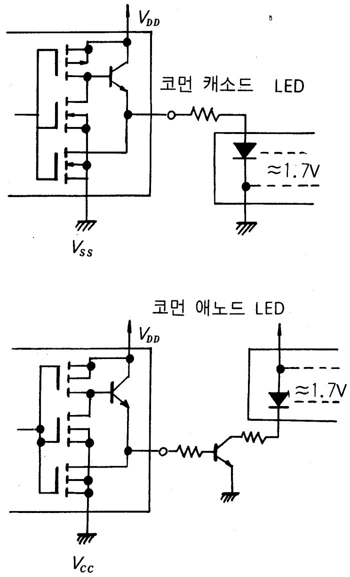
번호	입	력			출					력				
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L

(b) 진리표

〔그림 17〕 SN7447의 핀접속도



〔그림 18〕 LED의 드라이브법

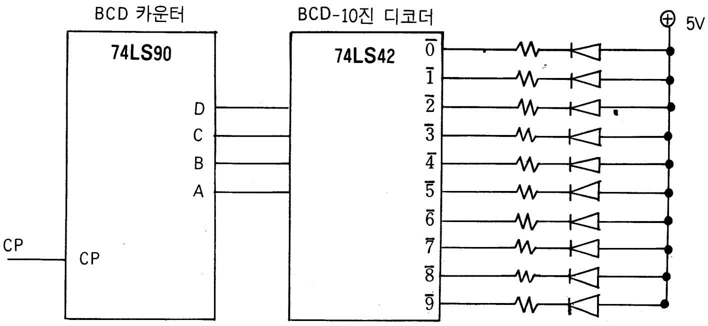


것으로 디지탈회로에서 빼 놓을 수 없는 회로이다. 카운터에는 플립플롭이 구성 요소로 사용되며 하나의 플립플롭회로는 2진 1자리의 계수를 할 수 있다. 따라서 플립플롭에 의한 카운터를 바이너리 카운터라고 한다.

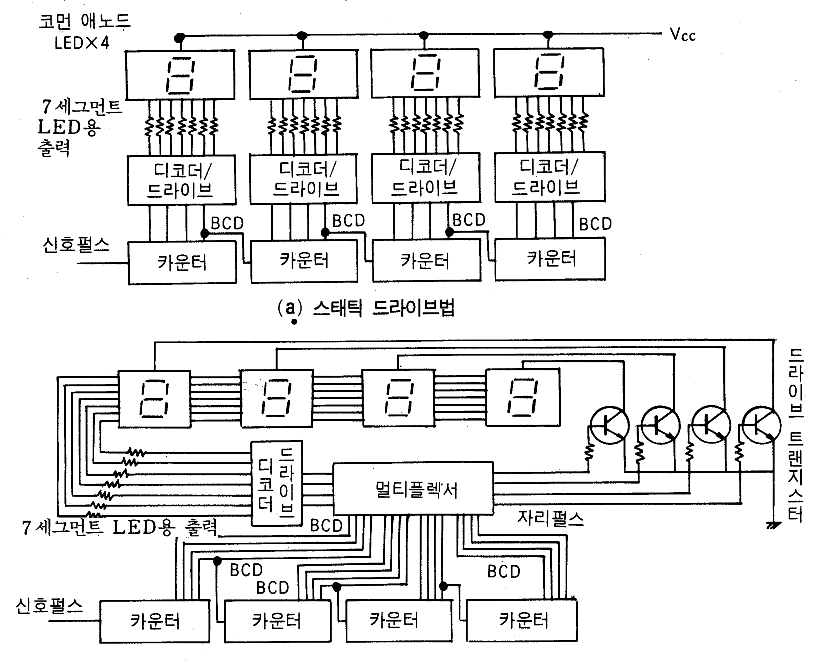
SN7490과 같이 0, 1, 2…의 방법으로 점점 수를 증가시켜 카운트하여 가는 것을 업 카운트라고 한다. 반대로 9, 8, 7과 같이 적은 쪽으로 카운트하는 것은 다운 카운터이다.

또 SN7490은 비동기식 카운터로, 카운터중의 FF그룹의 신호를 주고 받

〔그림 19〕 다이나믹 드라이브

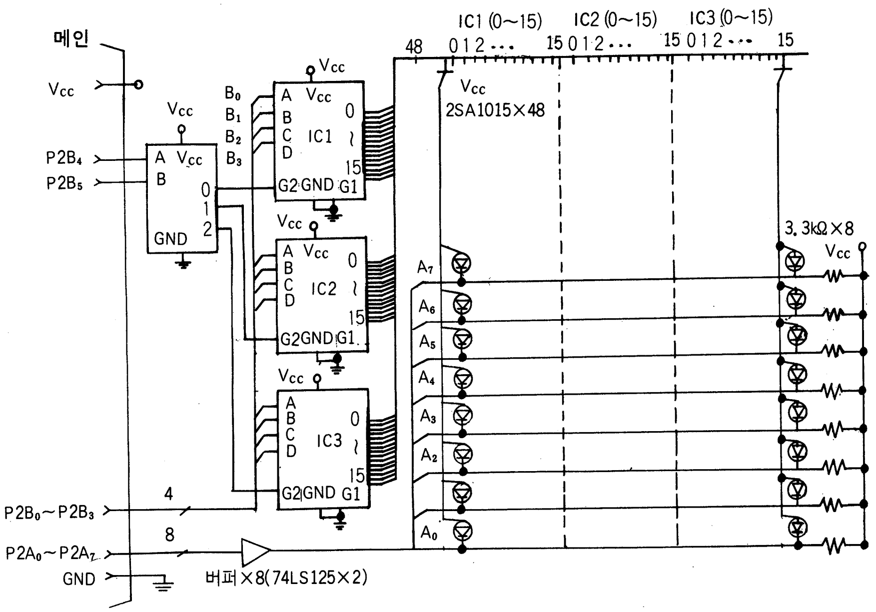


〔그림 20〕



(b) 다이나믹 드라이브법

〔그림 21〕 LED 드라이브의 회로도



을 때 약간 지연된다. 특히, 카운터IC를 여러단 직렬접속한 경우, 뒤의 카운터일수록 신호의 지연이 발생하여 때로는 불필요한 스파크가 생겨 오동작을 일으키는 경우가 있다.

이 결점을 보완하는 방법으로는 클럭펄스가 들어가지 않으면 카운트를 하지 않고 데이터이외의 불필요한 펄스가 들어가도 카운트하지 않는 동기식 카운터를 사용하는 방법이 있으나, 회로가 약간 복잡해져서 이번 제작에서는 피했다.

카운터에 대해 이해가 되었으면 그림 22의 회로도를 보자. 시각을 나타내기 위하여 7세그먼트 LED를 사용하였으나, PIO로 들어가는 입력은 2진수 데이터이어야 하므로 시중에서 구하기 쉬운 디지탈 클럭을 사용하지 않고, 비록 오차는 있더라도 NE555A를 이용하여 1Hz의 클럭을 발생시켜 이것을 카운트하는 회로를 만들었다.

여기서 발생하는 오차는 발진 회로내에 있는 반고정 저항을 사용하여 최소한으로 줄여보았다. 백업전원을 이용하지 않았기 때문에 전원을 일시 중단하였다 재공급하는 경우는 인위적으로 데이터를 수정할 수 있도록 푸시버튼 스위치를 2개 달았다. 또한 PIO 3의 A포트에 '분'을, B포트에 '시'를 입력받도록 하였다.

SN74125(반도체 스위치)를 사용한 이유는 시각의 리세트에 있어서 '0AH'일 때와 '25H'일 때 리세트를 걸어야 하고, '0AH'일 때는 10자리에 클럭도 입력되어야 하는 필요성을 충족시키기 위함이었다.

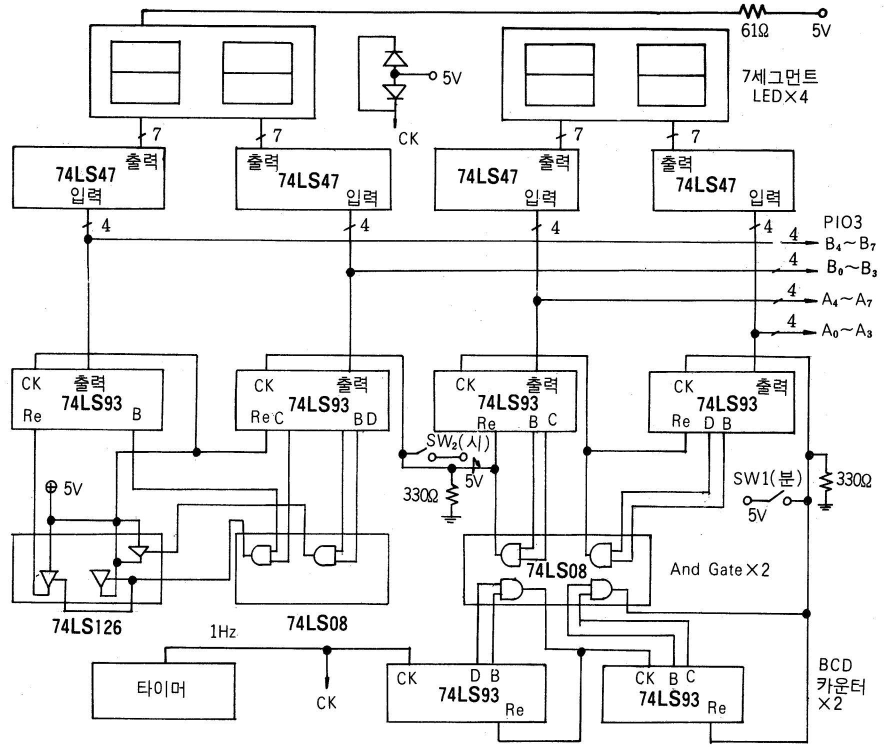
시간을 임의로 수정할 때에는 '분'을 먼저 조정하면 안된다. 항상 '시'를 맞추어 놓고 그 다음에 '분'을 맞추어야 한다. 이유는 '시'의 클럭이 '분'의 리세트에 연결되어 있기 때문이다.

시계에 대해서는 시중에서 구하기 쉬운 디지탈 시계를 사용해도 되는데, 이 때에는 PIO 2개를 사용해야 한다. 4개의 포트에 각각 시 10자리, H 1자리, 분 10자리, 분 1자리를 입력시키고 아울러 프로그램의 일부를 수정하여야 하는데 이것은 무척 간단한 문제이다.

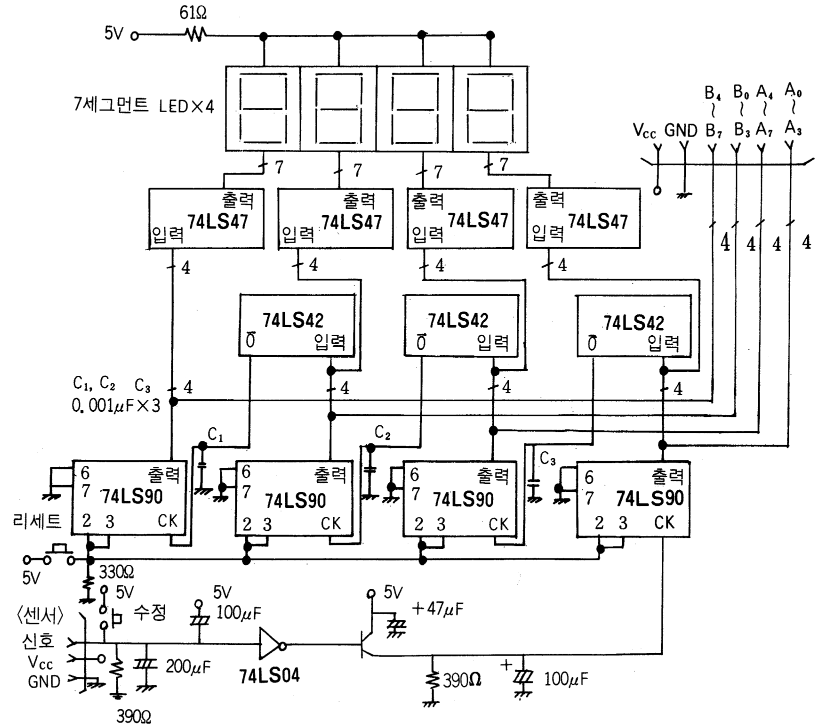
카운터

그림 23의 카운터회로는 센서에서 나온 검출 신호의 수를 세어 얼마만큼의 사람이 방문을 했는지 알 수 있는 회로이다. 이의 기본원리는 매우 간단하

〔그림 22〕 디지탈 시계의 회로도



〔그림 23〕 카운터회로



*각 IC의 Vcc와 GND는 생략되었음.

다. 즉 센서에서 나오는 검출신호가 채터링 제거회로를 거쳐서 나온 일정한 펄스를 SN7490의 입력단자에 접속시키고, 카운터에서 나오는 출력을 디코더를 통해 7세그먼트 LED로 보내는 것이다.

그리고 이 회로에 수정이 가능하도록 리세트스위치와 수정스위치를 첨부하였다. 전원을 넣었을 때는 리세트스위치를 눌러 모두 '0'으로 클리어시켜 놓고 작동시켜야 한다.

센서

이번 제작에서 가장 어려웠던 것이 이 센서부분이다. 이 부분에서 여러번 실패를 거듭하였다. 먼저 센서에 대한 기본원리부터 알아보자.

생물이든 무생물이든 간에 절대온도(-273℃)보다 높은 온도를 가지고 있는 한 그 온도에 해당하는 복사선을 방출하게 된다. 이 중 체온과 같이 온도가 낮은 것에서는 적외선이 복사된다. 물체로부터의 적외선 에너지를 전기신호로 변환하는 것이 바로 적외선 센서이다. 적외선 센서는 강도가 높고 응답속도가 빠른 등의 장점을 가졌으나 고가이며, 액체 질소 등으로 냉각시킬 필요가 있는 양자형 센서, 더미스터와 같이 DC적 동작을 하는 것과 초전형 센서와 같이 미분형 동작을 하는 것으로 나눠지는 열형 센서로 구분할 수 있는데 필자는 이번 제작에 사람의 움직임을 포착하는데 가장 좋은 초전형 센서를 사용하였다.

초전형 적외선 센서의 기본 동작을 간단히 설명하자면 다음과 같다. 초전형 적외선 센서의 주재료인 초전체는 재료 자체가 자발분극(Spontaneous Polarization : 전계를 가하지 않은 상태에서 유전체에 형성되어 있는 분극)을 가지고 대기중의 부유전하에 의해 중화되어 전위차는 없다. 그런데 적외선이 입사하여 초전체의 온도가 변화하면 초전효과가 일어나 전극간에 전위

차가 발생한다. 이러한 초전효과가 사람의 움직임을 포착해준다. 초전형 적외선 센서의 내부구조는 **그림 24**에 나타냈다. 이 초전형 적외선 센서를 2개 이용하여 한쪽 방향으로 이동하는 물체를 검출하는 회로는 **그림 25**와 같이 만들 수 있다. 2개의 센서 배치간격은 검출물체의 크기에 비하여 충분히 짧게 하여야 한다.

여기서는 대략적으로 4~5㎝정도의 간격을 두었다. A방향으로 물체가 이동하는 경우는 먼저 포토 트랜지스터 Tr_2를 차광하여 OFF된 후에 Tr_1을 OFF한다. 그리고 그 물체가 이동하여 Tr_1을 다 지나가면 그 순간 ⑤에서 검출신호가 나온다. B방향으로 물체가 이동하는 경우는 앞의 경우와 반대이다.

〔그림 24〕 초전형 적외선 센서의 내부회로 구조

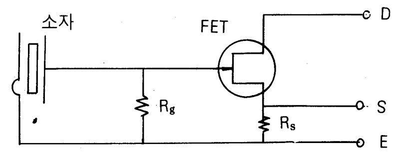


그림 25의 회로 ④, ⑤에서 나오는 펄스를 프리 셋 카운터에 연결시키면 전시장내에 있는 사람의 총인원을 알 수 있다. 그러나, 필자는 방문자의 총 인원수를 알기 위해서 한 방향 펄스만 이용하였다. 따라서 출입구같은 곳에서 들어오는 사람이나 나가는 사람중에 한 경우를 감지하고자 하는 경우에 무척 유용한 회로라고 생각한다.

이 때 포토 트랜지스터에서 나오는 검출신호에 스피커를 부착시키면 한번 물체를 감지할 때마다 소리를 낼 수도 있고, 그 밖에도 이 회로를 응용하여 도난경보기와 같은 것으로도 사용할 수 있다. 여기서 검출된 신호는 채터링 제거회로를 거쳐 SN7490의 입력단자에 들어간다.

프로그램

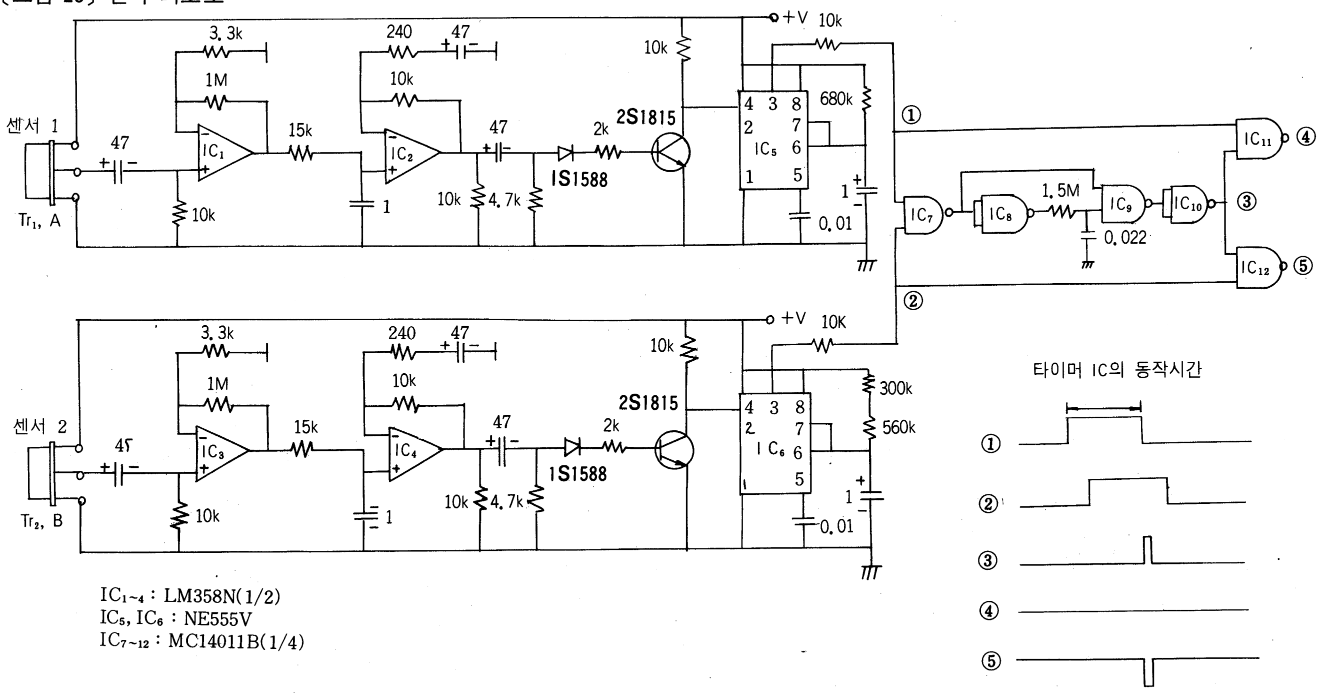
프로그램은 확장된 ROM 즉, 4000H 번지부터 들어가 있다. 프로그램을 보면 글자는 4150H 번지부터 **그림 26**과 같이 16진수로 처리하여 미리 입력시켜 놓았다. 데이터의 출력은 HL레지스터에 시작번지를 잡아주면 인덱스레지스터가 시작번지부터 끝번지까지(선두번지+30H(48)), B레지스터가 00H부터 30H까지 변하면서 데이터를 48번 출력한다. 그 다음에는 HL레지스터에 저장된 값, INC 또는 DEC에 의해 화면에 왼쪽 또는 오른쪽으로 흐르는가를 선택하게 된다.

GG01은 왼쪽, GG02는 오른쪽 방향을 선택한 것이다. 그 다음은 800EH 번지에 써 넣은 데이터에 따라 RRC, RLC, SCA를 선택하여 SUB 4에서 실행시킨다. 이것은 데이터를 위·아래로 이동시키는 작용을 한다.

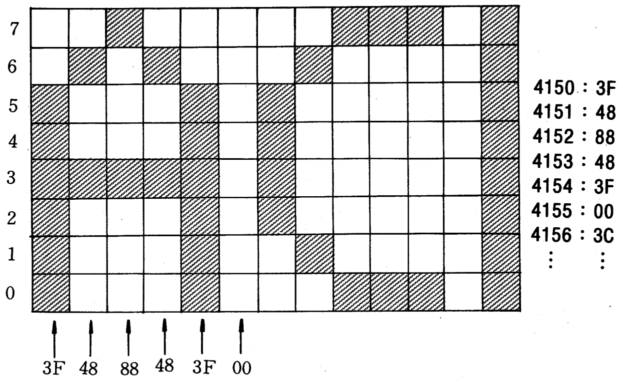
ROM에서는 데이터를 라이트시킬 수 없으므로 이 부분의 프로그램 40BBH 번지부터 40DB까지의 프로그램을 8000H 번지로 복사시킨다.

그 다음 시각과 방문자 수도 마찬가지로 문장내에 써 넣기 위하여 RAM으로 그 문장을 옮겨 놓고 시각 인원수 부분에 데이터를 써 넣음으로서 출

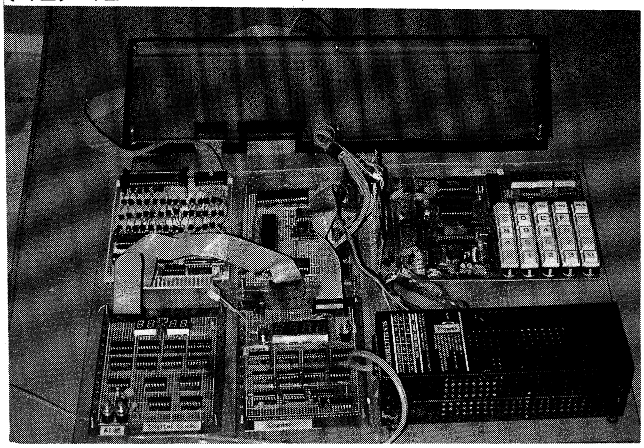
〔그림 25〕 센서 회로도



〔그림 26〕 데이터의 입력방법



〔사진〕 사인보드 내부구조



력을 하도록 하였다.

8256H~8283H까지에 시각을, 81ADH ~81CAH까지 카운트된 인원을 복사해 넣는다.

복사하는 방법은 PIO로부터 A에 받아들인 데이터를 SUB 5 루틴에서 상하위 4비트씩을 얻어 H에 49를 입력시키고 L에 4비트를 입력하면 4비트에 해당하는 10진수 데이터가 있는 곳의 시작하위번지를 얻어 다시 L에 입력하면 그 HL번지부터 +07H번지까지를 원하는 위치에 복사하는 방법을 썼다.

물론 미리 4900H번지에는 0에 해당하는 데이터가 있는 시작하위번지를, 4905H번지에는 5에 해당하는 데이터가 있는 시작하위번지를 설정하고 있다. 이와 같이 시작번지부터 그 수에 해당하는 포트를 미리 잡아 넣어 놓았다. 자세한 것은 리스트 1의 프로그램을 참고하기 바란다.

맺음말

제작하는 동안 회로를 여러번 수정하기도 하였다. 이는 먼저 회로에 문제가 있었음이고, 그 문제점을 해결하는 가운데서 더욱 많은 것을 배웠다고 스스로 평가해 본다.

이번 제작을 통해 하드웨어가 어렵게만 느껴지는 분들에게 마이크로컴퓨터를 정복할 수 있는 기회가 되었길 바란다.

비록 조그마한 사인보드이지만 얼마든지 확장시켜 기온, 습도, 소음정도, 인구 수 등 무수한 메시지를 내보낼 수가 있다. PC

〔리스트 1〕

```
        2500 A.D. Z80 CROSS ASSEMBLER  -  VERSION 3.01a
        -------------------------------------------------

              INPUT FILENAME : SIGN.ASM
              OUTPUT FILENAME : SIGN.OBJ

                         ;
                         ;        작동 PROGRAM
                         ;
                         ;        최종 수정일 : 1988.12.18
                         ;
   4000                           ORG  4000H
                         ;
   4000   01 20 00       START:   LD   BC,0020H
   4003   11 00 80                LD   DE,8000H
   4006   21 BB 40                LD   HL,40BBH
   4009   ED B0                   LDIR              ; DATA COPY
                         ;
   400B   01 00 03                LD   BC,0300H
   400E   11 67 80                LD   DE,8067H
   4011   21 67 45                LD   HL,4567H
   4014   ED B0                   LDIR              ; DATA COPY
                         ;
   4016   FD 21 D1 80             LD   IY,80D1H
   401A   3E 0F                   LD   A,0FH        ; PIO 2
   401C   D3 05                   OUT  (05H),A      ; 전 BYTE 출력 지정
   401E   D3 07                   OUT  (07H),A
                         ;
   4020   21 50 41                LD   HL,4150H
   4023   FD 36 00 40             LD   (IY+00H),40H
                         ;
   4027   E5             GG01:    PUSH HL           ; UNICOSA 관련
   4028   DD E1                   POP  IX           ; DATA 출력
   402A   CD DB 40                CALL SUB1
   402D   23                      INC  HL
   402E   3E 45                   LD   A,45H
```

```
4030  BC                  CP   H
4031  C2 27 40            JP   NZ,GG01
4034  3E 97               LD   A,97H
4036  BD                  CP   L
4037  C2 27 40            JP   NZ,GG01
403A  FD 36 08 80         LD   (IY+08H),80H
                    ;
403E  E5          GG02:   PUSH HL              ; DISPLAY
403F  DD E1               POP  IX
4041  CD DB 40            CALL SUB1
4044  2B                  DEC  HL
4045  3E 45               LD   A,45H
4047  BC                  CP   H
4048  C2 3E 40            JP   NZ,GG02
404B  3E 67               LD   A,67H
404D  BD                  CP   L
404E  C2 3E 40            JP   NZ,GG02
4051  CD FB 40            CALL SUB3
                    ;
4054  21 67 80            LD   HL,8067H        ; PRC
4057  3E 06               LD   A,06H           ; RLC
4059  32 0E 80            LD   (800EH),A       ; SLA
405C  CD 00 80            CALL 8000H           ; 선택
405F  3E 0E               LD   A,0EH
4061  32 0E 80            LD   (800EH),A
4064  CD 00 80            CALL 8000H
4067  3E 26               LD   A,26H
4069  32 0E 80            LD   (800EH),A
406C  FD 36 00 EF         LD   (IY+00H),0EFH
4070  16 0A               LD   D,0AH
4072  CD 06 80            CALL 8006H
                    ;
4075  3E 4F               LD   A,4FH           ; PIO 3,4
4077  D3 09               OUT  (09H),A         ; 전 BYTE 입력 지정
4079  D3 0B               OUT  (0BH),A
407B  D3 0D               OUT  (0DH),A
407D  D3 0F               OUT  (0FH),A
                    ;
407F  21 00 49            LD   HL,4900H
4082  DB 08               IN   A,(08H)         ; 각 part로 수치 입력
4084  11 73 82            LD   DE,8273H        ; 그 숫자 복사
4087  CD 25 41            CALL SUB5
408A  DB 0A               IN   A,(0AH)
408C  11 56 82            LD   DE,8256H
408F  CD 25 41            CALL SUB5
4092  DB 0C               IN   A,(0CH)
4094  11 BA 81            LD   DE,81BAH
4097  CD 25 41            CALL SUB5
409A  DB 0E               IN   A,(0EH)
409C  11 AD 81            LD   DE,81ADH
409F  CD 25 41            CALL SUB5
40A2  21 D0 80            LD   HL,80D0H
                    ;
40A5  E5          GGG:    PUSH HL              ; COUNTER 수치
40A6  DD E1               POP  IX              ;  CLOCK 수치 DISPLAY
40A8  CD DB 40            CALL SUB1
40AB  23                  INC  HL
40AC  3E 82               LD   A,82H
40AE  BC                  CP   H
40AF  C2 A5 40            JP   NZ,GGG
40B2  3E B0               LD   A,0B0H
40B4  BD                  CP   L
40B5  C2 A5 40            JP   NZ,GGG
40B8  C3 00 40            JP   START
                    ;
40BB  FD 36 00 80         LD   (IY+00H),80H    ; ROM에서
40BF  16 28               LD   D,28H           ;  RAM으로 옮겨
40C1  E5                  PUSH HL              ;  [ ]에 서 넣은
40C2  DD E1               POP  IX              ;  수치에 따라
40C4  06 00               LD   B,00H           ;    RRC
40C6  DD CB 00 0E         RRC  (IX+00H)        ;    RLC
40CA  DD 7E 00            LD   A,(IX+00H)      ;    SLA
40CD  CD E7 40            CALL SUB2            ;    작동
40D0  C2 06 80            JP   NZ,8006H
40D3  15                  DEC  D
40D4  C2 06 80            JP   NZ,8006H
40D7  CD FB 40            CALL SUB3
40DA  C9                  RET
                    ;
40DB  06 00       SUB1:   LD   B,00H           ; IX--IX+30 분의
40DD  DD 7E 00    NEXT:   LD   A,(IX+00H)      ; DATA를 한번에 출력
40E0  CD E7 40            CALL SUB2
40E3  C2 DD 40            JP   NZ,NEXT
40E6  C9                  RET
                    ;
40E7  EE FF       SUB2:   XOR  0FFH            ; DATA 반전
40E9  D3 04               OUT  (04H),A         ; A PORT DATA
40EB  78                  LD   A,B             ; B PORT ADDRESS
40EC  D3 06               OUT  (06H),A         ; 출력
40EE  FD 4E 00            LD   C,(IY+00H)
                    ;
40F1  0D          LOOP:   DEC  C
40F2  20 FD               JR   NZ,LOOP
40F4  DD 23               INC  IX
40F6  04                  INC  B
40F7  78                  LD   A,B
40F8  FE 30               CP   30H
40FA  C9                  RET
                    ;
40FB  FD 36 00 80 SUB3:   LD   (IY+00H),80H    ; DATA 5번
40FF  16 05               LD   D,05H           ; 깜박거림
                    ;
4101  1E 05       GG06:   LD   E,05H
                    ;
```

```
4103  E5        GGO3:  PUSH HL
4104  DD E1            POP  IX
4106  CD DB 40         CALL SUB1
4109  1D               DEC  E
410A  C2 03 41         JP NZ,GGO3
410D  1E 05            LD   E,05H
                ;
410F  E5        GGO5:  PUSH HL
4110  DD E1            POP  IX
4112  06 00            LD   B,00H
                ;
4114  3E 00     GGO4:  LD   A,00H
4116  CD E7 40         CALL SUB2
4119  C2 14 41         JP NZ,GGO4
411C  1D               DEC  E
411D  C2 0F 41         JP NZ,GGO5
4120  15               DEC  D
4121  C2 01 41         JP NZ,GGO6
4124  C9               RET
                ;
4125  FD 77 00  SUB5:  LD   (IY+00H),A   ; COUNTER
4128  E6 F0            AND  0F0H        ; CLOCK
412A  0E 04            LD   C,04H       ;  숫자 입력 받아
412C  CB 0F     GGO:   RRC  A           ;   7 BYTE의 DATA로
412E  0D               DEC  C           ;    바꿔 각 위치에
412F  C2 2C 41         JP NZ,GGO        ;      복사함
4132  6F               LD   L,A
4133  7E               LD   A,(HL)
4134  6F               LD   L,A
4135  01 07 00         LD   BC,0007H
4138  ED B0            LDIR
413A  FD 7E 01         LD   A,(IY+01H)
413D  E6 0F            AND  0FH
413F  6F               LD   L,A
4140  7E               LD   A,(HL)
4141  6F               LD   L,A
4142  01 07 00         LD   BC,0007H
4145  ED B0            LDIR
4147  C9               RET
4148                   END
```

〈용어해설〉

●디스플레이(Display)

입력된 데이터를 CRT를 통해 출력하는데 있어서 도형이나 문자, 숫자 등과 같은 형태를 사용해 시각적으로 표현되는 현상을 말한다. 디스플레이방법으로는 음극선관을 사용한 음극선관표시, 플라즈마표시, EL표시, 액정표시 등이 있는데 최근에는 마이크로프로세서를 장착한 디스플레이장치가 많이 보급되어 다양한 표시가 가능하다.

●스위칭 레귤레이터(Switching Requlater)

안정화전원의 하나로서 트랜지스터와 같은 스위칭소자를 사용하여 직류의 입력전압이 인가되었을 경우, 직류의 출력전압을 얻을 수 있도록 입력전압을 고속으로 스위칭하고, 그로 인해 얻어진 펄스열을 정류하고 평활한다. 이 전원은 디지탈회로에 주로 사용된다.

●원보드 컴퓨터(One Board Computer)

일반적으로 1개의 LSI칩에 집적된 원칩 CPU를 마이크로프로세서라고 부르고, 이 마이크로프로세서에 메모리를 부가한 것이 원보드 컴퓨터이다. 여기에 다시 콘솔(Console) 패널을 붙여서 케이스에 넣고 전원을 조립한 것이 마이크로컴퓨터이다.

●콘솔(Console)

제어용 키, 스위치, 지시기 등을 갖춰 오퍼레이터가 필요에 따라 컴퓨터의 동작에 개입하거나 감시할 수 있도록 하는 터미널

●레지스터(Register)

1비트 또는 복수비트를 유지하는 기억장치나 회로로서 CPU내부에서 각종 명령을 실행할 때, 그 명령 내용이나 데이터를 일시적으로 축적하여 필요에 따라 수시로 그 내용을 사용할 수 있도록 되어 있다.

●인터럽트(Interrupt)

특정 사건이 발생했을 경우, 하드웨어적인 수단으로 현재 실행되고 있던 프로그램을 일시 중단시키고, 그 발생된 사건에 해당되는 프로그램을 먼저 실행시킨 뒤 그 실행된 프로그램이 완료되면 다시 원래 CPU에서 실행되던 프로그램으로 복원하여 그 프로그램을 실행하는 현상을 말한다. 인터럽트에는 입출력장치의 개시, 종료인터럽트, 내부인터럽트, 외부인터럽트, 기계체크 인터럽트 등이 있다.

●핸드세이킹(Handshaking)

출력되는 데이터에 관계없이 연속적으로 입력데이터를 전송할 경우 현재 입력된 데이터가 출력되기도 전에 그 다음 입력 데이터가 들어옴으로써 잘못된 출력을 얻는 경우가 생긴다. 이를 방지해 주는 역할을 하는 것이 핸드셰이크로서 이는 제어신호를 사용하며 송신과 수신측에서 서로 데이터를 주고 받는데 별무리가 없는지를 확인하면서 데이터를 전송한다.

●정격전류(Rated Current)

정격온도에서 설계상 교류용 캐패시터에 인가할 수 있는 정격주파수의 교류주파수의 실효값 즉 $I2\pi fcv \times 10^{-6}$[A]이다.

●채터링(Chattering)

스위치나 릴레이 등의 접점이 개폐하는 경우에 기계적인 원인에 의해서 생기는 불규칙한 진동

삼보컴퓨터의 파킹 프로그램
TGSHIP.COM의 분석

최명환

TGSHIP.COM은 삼보컴퓨터에서 제공되는 하드디스크의 파킹 프로그램이다. 파크(Pask)라는 말이 '주차하다', '정박하다', '쉬다' 등의 의미를 지니고 있듯이 컴퓨터를 사용하고 난 후, 스위치를 OFF시키기 전에 파킹 프로그램을 실행시켜 주면 하드디스크 드라이브의 헤드를 안전한 곳으로 이동시켜 하드디스크를 안전하게 유지시킬 수가 있는 것이다.

파킹 프로그램을 실행시키지 않고 컴퓨터의 전원을 OFF시키게 되면 드라이브의 헤드가 마지막으로 데이터를 읽던 부분에 그대로 위치한 채로 있다가 외부에서 가해지는 충격에 의해 디스크면에 손상을 입힐 수가 있기 때문에 대부분의 하드디스크 생산업체에서는 나름대로의 파킹 프로그램들을 제공하고 있다. 필자는 삼보컴퓨터의 TGSHIP.COM을 이용하고 있는데 이 프로그램의 장점은 헤드를 파크시킨 후에도 계속 프롬프트가 남아 있는채로 있다가 원하면 작업을 계속할 수 있도록 되어 있기 때문이다(화면 1 참조). (다른 경우로써 SD.EXE인 경우는 일단 실행시키면 시스템이 정지상태로 되어 버리기 때문에 다시 작업을 하고 싶으면 스위치를 OFF시켰다가 다시 켜야 하는 단점이 있다.)

이렇게 기존의 프로그램을 힘들여 역어셈블하게 된 동기는 별다른 이유가 있어서가 아니라 필자도 마찬가지이지만 파킹 프로그램에 대한 호기심과 궁금증을 갖고 있는 독자들에게 어느 정도 도움이 됐으면 하는 바램에서이다.

리스트 1에 전체적인 소스를 보였는데 앞부분 즉 오프셋 0103번지부터 0498번지까지는 전부 화면을 구성하는 데이터들로 채워져 있음을 볼 수 있다. 여기서 한가지 눈여겨 보아야 할 것은 두번째 스트링이 시작되는 곳의 0253번지와 0262~0265번지에 '00'이 채워져 있는데 나중에 프로그램이 실행되면서 0253번지에는 체크할 드라이브('C' 또는 'D')명이, 0262~0265번지에는 체크되어진 하드디스크의 실린더 수가 들어가게 된다는 점이다.

```
MOV  DX,0080H
MOV  BYTE PTR [SCR_DATA_BEGIN+150H],43H
:        :
MOV  DX,0081H
MOV  BYTE PTR [SCR_DATA_BEGIN+150H],44H
:        :
```

화면 1의 예를 보면 드라이브명으로 'C'가, 실린더 수로 614가 출력되고 있음을 볼 수 있는데 이 마지막 실린더, 즉 디스크면의 제일 안쪽으로 헤드가 이동되는 것이다.

하드디스크 드라이브의 실린더 수를 구하는 부분은 라벨 L04D6 : 으로 시작되는 곳인데 이 부분을 잠시 살펴보면 다음과 같다.

```
      L04D6:
MOV   BP,DX
MOV   AH,08H
INT   13H
JB    L04CB
:      :
```

'INT 13H / 08H'는 드라이브의 파라미터를 구하는 기능을 가지며 그 출력으로 CX레지스터에는 최대 트랙수와 트랙당 최대 섹터수가, DX레지스터에는 최대 헤드수와 드라이브 수가 구해지게 된다(그림 1 참조).

그리고 하드디스크를 발견하지 못하면 'JB L04CB'에 의해 라벨 L04CB : 로 점프해서 나머지 화면 데이터를 출력하고 프로그램을 종료시킨다.

실제로 헤드가 지정된 트랙(실린더)의 위치로 이동하는 것은 오프셋 04F2번지부터 시작되는 다음의 명령에 의해서 이루어진다.

〔화면 1〕 TGSHIP을 실행시킨 예

```
The SHIP Utilities  V3.21 (C) TriGem Computer Inc.1987

    Moving head of drive C: to Cylinder  614

 ****************** Warning *******************
 * Be sure to turn OFF the system power switch  *
 * before shipping this hard_disk !!!           *
 ***********************************************

C:\>
```

〔리스트 1〕 TGSHIP.COM의 소스

```
          ;         TGSHIP --- TESTIFIE
          ;         ------ 1989.APRIL FOOLS DAY -------
          CODE      SEGMENT
                    ASSUME CS:CODE
                    ORG 100H
                    PROGRAM_START   PROC
     BEGIN:         JMP     START ;->0100 E99603  [JMP 499]

CR      =       0DH              ;RETURN
LF      =       0AH              ;LINE FEED
     SCR_DATA_BEGIN:
DB   '╔════════════'        ;->0103      C9 CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->0110  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->0120  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═════════════╗',CR,LF   ;->0130  CD CD CD CD CD CD CD CD-CD CD CD CD CD BB 0D 0A
DB '║ The SHIP Utili'       ;->0140  BA 20 54 68 65 20 53 48-49 50 20 55 74 69 6C 69
DB 'ties  V3.21 (C) '       ;->0150  74 69 65 73 20 20 56 33-2E 32 31 20 28 43 29 20
DB 'TriGem Computer '       ;->0160  54 72 69 47 65 6D 20 43-6F 6D 70 75 74 65 72 20
DB 'Inc.1987  ║',CR,LF,'╠═' ;->0170  49 6E 63 2E 31 39 38 37-20 20 BA 0D 0A CC CD CD
DB '═══════════════'        ;->0180  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->0190  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->01A0  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════╣',CR,LF,'║    ' ;->01B0  CD CD CD CD CD CD CD B9-0D 0A BA 20 20 20 20
DB '               '        ;->01C0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->01D0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->01E0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '    ║',CR,LF,'║     '   ;->01F0  20 20 20 20 BA 0D 0A BA-20 20 20 20 20 20 20 20
DB '               '        ;->0200  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0210  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0220  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB ' ║',CR,LF,'$║      Mo'  ;->0230  20 BA 0D 0A 24 BA 20 20-20 20 20 20 20 20 4D 6F
DB 'ving head of dri'       ;->0240  76 69 6E 67 20 68 65 61-64 20 6F 66 20 64 72 69
DB 've : to Cylinde'        ;->0250  76 65 20 00 3A 20 74 6F-20 43 79 6C 69 6E 64 65
DB 'r          ║'          ;->0260  72 20 00 00 00 00 20 20-20 20 20 20 20 20 20 BA
DB '',CR,LF,'$║     '       ;->0270  0D 0A 24 BA 20 20 20 20-20 20 20 20 20 20 20 20
DB '               '        ;->0280  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0290  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '          ║',CR,LF      ;->02A0  20 20 20 20 20 20 20 20-20 20 20 20 20 BA 0D 0A
DB '║              '        ;->02B0  BA 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
DB '               '        ;->02C0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->02D0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '       ║',CR,LF,'║ '    ;->02E0  20 20 20 20 20 20 20 20-20 20 BA 0D 0A BA 20 20
DB '   *************'       ;->02F0  20 20 20 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A
DB '****** Warning *'       ;->0300  2A 2A 2A 2A 2A 2A 20 57-61 72 6E 69 6E 67 20 2A
DB '****************'       ;->0310  2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A
DB '***    ║',CR,LF,'║    ' ;->0320  2A 2A 2A 20 20 20 20 BA-0D 0A BA 20 20 20 20 20
DB '* Be sure to tur'       ;->0330  2A 20 42 65 20 73 75 72-65 20 74 6F 20 74 75 72
DB 'n OFF the system'       ;->0340  6E 20 4F 46 46 20 74 68-65 20 73 79 73 74 65 6D
DB ' power switch  *'       ;->0350  20 70 6F 77 65 72 20 73-77 69 74 63 68 20 20 2A
DB '    ║',CR,LF,'║    * b' ;->0360  20 20 20 20 BA 0D 0A BA-20 20 20 20 20 2A 20 62
DB 'efore shipping t'       ;->0370  65 66 6F 72 65 20 73 68-69 70 70 69 6E 67 20 74
DB 'his hard_disk !!'       ;->0380  68 69 73 20 68 61 72 64-5F 64 69 73 6B 20 21 21
DB '!          *   '        ;->0390  21 20 20 20 20 20 20 20-20 20 20 20 2A 20 20 20
DB ' ║',CR,LF,'║   ******'  ;->03A0  20 BA 0D 0A BA 20 20 20-20 20 2A 2A 2A 2A 2A 2A
DB '****************'       ;->03B0  2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A
DB '****************'       ;->03C0  2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A
DB '**********     ║.'      ;->03D0  2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 20 20 20 20 BA 0D
DB '.║              '       ;->03E0  0A BA 20 20 20 20 20 20-20 20 20 20 20 20 20 20
DB '               '        ;->03F0  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0400  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '          ║',CR,LF,'║ ' ;->0410  20 20 20 20 20 20 20 20-20 20 20 BA 0D 0A BA 20
DB '               '        ;->0420  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0430  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '               '        ;->0440  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20
DB '        ║',CR,LF,'╚══'  ;->0450  20 20 20 20 20 20 20 20-20 BA 0D 0A C8 CD CD CD
DB '═══════════════'        ;->0460  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->0470  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═══════════════'        ;->0480  CD CD CD CD CD CD CD CD-CD CD CD CD CD CD CD
DB '═════╝',CR,LF,'$'       ;->0490  CD CD CD CD CD BC 0D 0A-24 B8
     START:
  MOV    AX,0600H                     ;->0499 B80006
  MOV    CX,0000H                     ;->049C B90000
  MOV    BH,07H                       ;->049F B707
  MOV    DX,174FH                     ;->04A1 BA4F17
  INT    10H                          ;->04A4 CD10
```

화면 데이터영역 (0103~0490)

화면을 클리어시킨다. (0499~04A4)

```
  XOR    DX,DX                        ;->04A6 33D2
  XOR    BH,BH                        ;->04A8 32FF
  MOV    AH,02H                       ;->04AA B402
  INT    10H                          ;->04AC CD10
```

커서를 화면의 좌측 상단에 위치시킨다 (X=00, Y=00).

```
  MOV    DX,0103H                     ;->04AE BA0301
  MOV    AH,09H                       ;->04B1 B409
  INT    21H                          ;->04B3 CD21
```

0103번지부터 0234번지까지의 화면 데이터 출력

```
  MOV    DX,0080H                     ;->04B5 BA8000
  MOV    BYTE PTR [SCR_DATA_BEGIN+150H],43H
  CALL   L04D6                        ;->04BD E81600
```

C드라이브 체크 루틴 호출

```
  MOV    DX,0081H                     ;->04C0 BA8100
  MOV    BYTE PTR [SCR_DATA_BEGIN+150H],44H
  CALL   L04D6                        ;->04C8 E80B00
```

D드라이브 체크 루틴 호출

```
         L04CB:
  MOV    DX,0273H                     ;->04CB BA7302
  MOV    AH,09H                       ;->04CE B409
  INT    21H                          ;->04D0 CD21
  MOV    AH,4CH                       ;->04D2 B44C
  INT    21H                          ;->04D4 CD21
```

0273번지부터 끝까지 화면 데이터 출력. 종료

```
         L04D6:
  MOV    BP,DX                        ;->04D6 8BEA
  MOV    AH,08H                       ;->04D8 B408
  INT    13H                          ;->04DA CD13
```

드라이브 파라미터 알아내기

```
  JB     L04CB                        ;->04DC 72ED
```

하드디스크가 없으면 프로그램 종료

```
  INC    CHH                          ;->04DE FEC5
  JNZ    L04EA                        ;->04E0 7508
  ADD    CL,40H                       ;->04E2 80C140
  JNB    L04EA                        ;->04E5 7303
  ADD    DH,40H                       ;->04E7 80C640
```

```
        L04EA:
DEC     CL              ;->04EA FEC9
MOV     BX,BP           ;->04EC 8BDD
MOV     DL,BL           ;->04EE 8AD3
PUSH    DX              ;->04F0 52     } 레지스터 저장
PUSH    CX              ;->04F1 51
MOV     AH,0CH          ;->04F2 B40C   } 트랙(실린더)
INT     13H             ;->04F4 CD13     찾기(헤드를 옮기는 부분)
POP     CX              ;->04F6 59     } 레지스터 복원
POP     DX              ;->04F7 5A
JB      L04CB           ;->04F8 72D1 ←── 하드디스크가 없으면
ROL     DH,1H           ;->04FA D0C6     프로그램 종료
ROL     DH,1H           ;->04FC D0C6
AND     DH,03H          ;->04FE 80E603
XCHG    CL,CH           ;->0501 86CD
RCL     CH,1            ;->0503 D0D5
RCL     DH,1            ;->0505 D0D6
RCL     CH,1            ;->0507 D0D5
RCL     DH,1            ;->0509 D0D6
MOV     CH,DH           ;->050B 8AEE
MOV     BX,CX           ;->050D 8BD9
MOV     CX,0010H        ;->050F B91000
XOR     DX,DX           ;->0512 33D2
        L0514:
ADD     BX,BX           ;->0514 03DB
MOV     AL,DL           ;->0516 8AC2
ADC     AL,AL           ;->0518 12C0
DAA                     ;->051A 27
MOV     DL,AL           ;->051B 8AD0
MOV     AL,DH           ;->051D 8AC6
ADC     AL,AL           ;->051F 12C0
DAA                     ;->0521 27
MOV     DH,AL           ;->0522 8AF0
LOOP    L0514           ;->0524 E2EE
MOV     BX,0262H        ;->0526 BB6202
```

(ROL DH,1H ~ LOOP L0514: 파라미터로부터 얻은 값 (CX, DX)을 실제 실린더의 값으로 변환하는 계산 루틴)

```
MOV     AL,DH           ;->0529 8AC6
CALL    L053B           ;->052B E80D00
MOV     AL,DL           ;->052E 8AC2
CALL    L053B           ;->0530 E80800
MOV     DX,0235H        ;->0533 BA3502
MOV     AH,09H          ;->0536 B409
INT     21H             ;->0538 CD21
RET                     ;->053A C3
        L053B:
PUSH    AX              ;->053B 50
SHR     AL,1            ;->053C D0E8
SHR     AL,1            ;->053E D0E8
SHR     AL,1            ;->0540 D0E8
SHR     AL,1            ;->0542 D0E8
CALL    L0548           ;->0544 E80100
POP     AX              ;->0547 58
        L0548:
AND     AL,0FH          ;->0548 240F
OR      AL,30H          ;->054A 0C30
CMP     BX,0262H        ;->054C 81FB6202
JNZ     L0558           ;->0550 7506
CMP     AL,30H          ;->0552 3C30
JNZ     L0558           ;->0554 7502
MOV     AL,20H          ;->0556 B020
        L0558:
MOV     [BX],AL         ;->0558 8807
INC     BX              ;->055A 43
RET                     ;->055B C3

    PROGRAM_START   ENDP
        CODE        ENDS
        END         BEGIN
```

〔그림 1〕 드라이브 파라미터 알아내기

```
MOV DX, 0080H
MOV AH, 08H
INT  13H
```

	High	Low
AX	08H	
BX		
CX		
DX		드라이브 번호

(a) 사용전

	High	Low
AX	00H	00H
BX	00H	드라이브 형태
CX	최대 트랙 수	트랙당 최대 섹터 수
DX	최대 헤드 수	드라이브 수

(b) 사용후

〔그림 2〕 트랙(실린더) 찾기

```
 :   :
MOV AH, 0CH
INT  13H
```

	High	Low
AX	0CH	
BX		
CX	트랙번호	
DX	헤드번호	드라이브번호

	High	Low
AX	상태코드	
BX		
CX		
DX		

※ 상태코드는 컴퓨터 관련 '디스크 시스템 상태코드'를 참고 바람

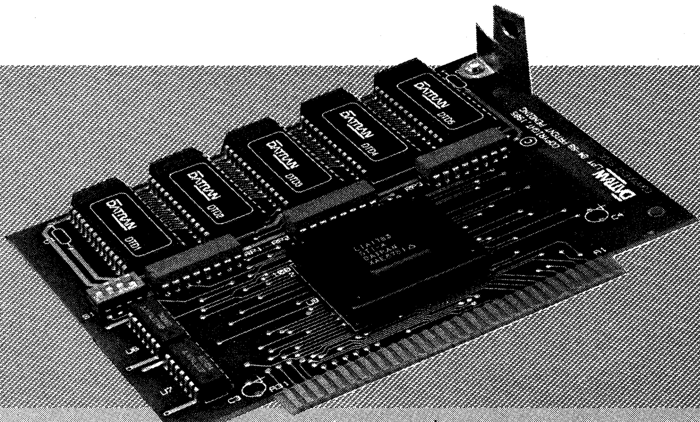
```
MOV  AH,0CH
INT  13H
 :    :
```

즉 CX와 DX레지스터에 트랙번호와 드라이브번호 및 헤드번호를 지정하고 INT 13H/0CH를 실행시키면 되는데 그림 2를 보면 쉽게 이해할 수 있을 것이다.

나머지 부분은 화면 데이터를 출력하거나 프로그램 종료 및 실린더수를 계산하는 루틴, 아스키형식으로 변환하는 루틴 등으로 이루어져 있다.

"Hot News" PC 이용 효율 극대화!

diskdoubler™ dd2000



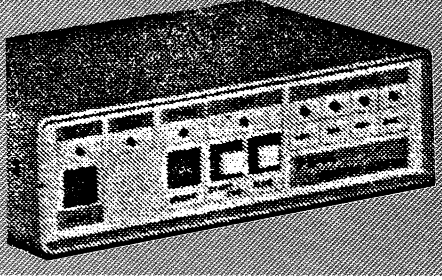
Disk Drive의 용량을 2배로!

- IBM-PC/XT, AT, 386, and PS/2에 사용가능
- 어떤 Disk Drives(Type, 용량불문)에도 적용(Floppy, ST-506/ESDI/SCSI Disk, CTD, Removable Disk, and optical Disk)
- Data(Files)의 자동압축-복귀가능
- 32MB DOS Barrier가 64MB로 확장
- Deluxe Packing with short card, S/W and Manual



Easy way to Link two Pcs!

- 단지 RS-232 Cable과 Software Driver만으로 two Pcs를 Network으로 연결
- 상대방의 Floppy, Hard Disk, and Printer를 Background(Multi tasking)공유
- 어떤 PC간에도 연결가능
  (Desk top(XT, AT) 상호간, Desk top-Laptop)
- Novell, Token Ring, and 3 com과 같은 Network에 접속하여 사용가능
- Deluxe Packing with cable, software, and Manual



Printer Buffer(Model PB-256-2)

- 용량 : 256KB
- Interface : centronics parallel
- 입출력 수 : 2 : 1 or 1 : 2
- 기능 : Self Test Copy (255장가지 가능) 및 Memory기능

*상기 품목의 취급점을 모집하고 있습니다.

*학생 및 개인용 PC & PRINTER 염가 판매

X T	AT	Printer
• 4.77/10MHz 8088-1 CPU • 640KB Memory • 2×5¼" 500KB Floppy Drive • Real Time Clock/Calendar • 8 Expansion Slots • 12" Green Monitor • 84Keys Keyboard (Hard Disk Option)	• 16MHz 80286-16 CPU • 1MB Memory • 1×5¼" 1.2MB Floppy Drive • Half Height 20MB Hard Disk • Serial/Parallel Port • 7 Expansion Slots • Battery Back Up Real Time Clock • 14" Green Monitor • 101 Keys Keyboard	**PS5123** • 24pin • 300CPS, 136COL • 7 color printer • Head 보호기능 • 조작 Pannel에서 사양 설정 * 낱장 공급장치는 Option * 9Pin, 80COL Printer도 공급가능

메리트시스템
MERIT
주소 : 서울특별시 강남구 역삼동 642-5(Rm.201)
전화 : 554-7919 FAX : 555-9106

판매협력점
용 산 : 나진상가 18동 가열 337-8호(701-5060)
충무로 : KSC Korea., LTD(274-3038)
수 원 : 바로크 전자(6-0114)

키보드(1)

키보드는 인텔 마이크로프로세서를 갖추고 있는데, 이 마이크로프로세서는 모든 키입력을 감지하여 시스템보드에 놓인 8255 주변 인터페이스칩의 포트 A에 스캔코드를 저장한다.

스캔코드는 1바이트로 구성되어 있고, 이 바이트의 하위 7비트는 각 키에 지정되어 있는 임의의 고유번호를 의미한다. 그리고 스캔코드의 최상위 비트는 키가 눌려졌는지(비트=1, 생성코드 : Make Code), 떼어졌는지(비트=0, 소멸코드 ; Break Code)를 나타낸다.

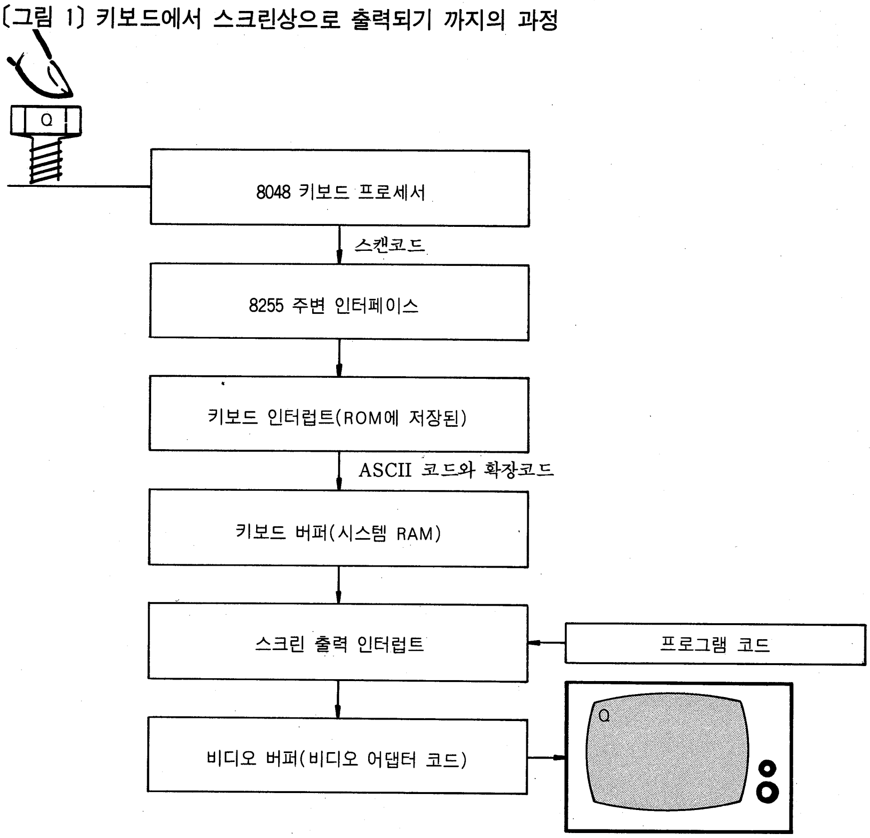
예를 들어 B키의 7비트 스캔코드는 48이고, 2진수로는 110000이므로 이 키를 누르면 포트 A에 전송되는 코드는 10110000이 되고, 키를 놓으면 이 코드는 00110000으로 된다. 따라서 모든 키입력은 8255칩에 2차례씩 기록된다. 그리고 매 기록마다 8255는 키보드의 마이크로프로세서로 "애크놀리지(Acknowledge : 긍정응답)" 신호를 보낸다. AT에서도 이와 마찬가지이나 매번 스캔코드 앞에 F0H바이트를 함께 보내주는 것이 약간 다르다.

스캔코드가 포트 A에 일시 저장되면 키보드 인터럽트(INT 9)가 걸리며, CPU는 순간적으로 모든 동작을 정지시키고 스캔코드를 분석하는 루틴을 수행한다. 이 스캔코드가 "시프트"나 "토글" 키로부터 발생되면 키의 상태 변화가 메모리에 기록되는데 이로써 스캔코드는 문자코드로 바뀌어 키를 누른 결과를 나타내게 된다. (그렇게 하지 않으면 이 스캔코드는 소멸된다) 물론 이 루틴은 문자코드가 올바른가("a"인가 또는 "A"인가)를 확인하기 위해 시프트 및 토글키의 상태를 우선적으로 체크한다.

그리고 난후 문자코드는 키보드 버퍼로 옮겨지는데 이 키보드 버퍼는 프로그램이 그들을 처리하느라 바쁠동안에도 들어오는 문자를 15개까지 저장시킬 수 있는 메모리용량을 갖고 있다. 그림 1은 키입력에서 입력한 내용이 스크린상에까지 전달되는 과정을 보인 것이다.

문자코드에는 ASCII코드와 확장코드(Extended Code) 등 2종류가 있다. ASCII코드는 IBM확장 ASCII문자에 대응하는 1바이트번호이다. IBM PC에서의 ASCII코드는 일반 타이프문자와 많은 특별문자 및 블록그래픽 기호를 포함하고 있다. ASCII코드는 이외에도 스크린상에서 문자로 사용되기보다는 주변장치로 명령어를 보내는데 사용되는 32개의 제어코드까지 포함하

〔그림 1〕 키보드에서 스크린상으로 출력되기 까지의 과정



고 있는데 이들 코드는 직접적인 메모리 맵핑에 의해 비디오 디스플레이상에 디스플레이시킬 수 있는 각각의 고유기호를 갖는다.

말하자면 반드시 처음 128문자만이 진정한 ASCII코드라고 하는 것은 무리이다. 왜냐하면 ASCII는 "Amrican Standard Code for Information Interchange"의 약자이기 때문이다. 그러나 프로그래머들은 보통 다른 번호와 구별하기 위해서 ASCII코드라 말한다. 예를 들면 ""ASCII 8"은 "백스페이스(Backspace)"문자를 가르키고 "8" 은 아스키코드 56에 대한 문자를 나타내게 되는 것이다.

확장코드는 키를 나타내는 ASCII기호를 갖고 있지 않는 함수키와 Alt키 결합에 의한 키 또는 키결합으로 설정된다.

확장코드는 2바이트 길이로 첫바이트는 항상 ASCII 0이며 두번째 바이트는 코드번호이다.

예를 들어 0:30 코드는 Alt-A를 나타낸다. 프로그램에 놓인 처음의 0는 코드번호가 ASCII코드인지 확장코드인지를 구별하여 준다.

스캔코드를 발생시키지 않으면서 특별한 기능을 수행하는 키결합이 일부 있는데, AT에서는 〈Ctrl-Break〉,〈Ctrl-Alt-Del〉, 〈PrtSc〉 및 〈SysReg〉 등이 그것이다. 그리고 모든 키입력은 프로그램에 의해서 해석되나 만일 커서를 왼쪽으로 이동시키는(Cusor-Left) 것과 같이 특별한 목적을 가진 경우에는 프로그램 자체에서 이 기능을 수행할 수 있는 코드를 부여해 주어야만 된다.

다행히도 OS는 모든 스트링을 한꺼번에 받아들일 수 있는 키보드버퍼에서 코드를 읽어낼 수 있을만큼 많은 루틴을 갖고 있다.

이 루틴은 유저가 요구할 수 있는 모든 것을 바로 실행해주기 때문에 유저 자신이 키보드 수행절차를 라이트시킬 필요는 없다.

키보드버퍼의 클리어

프로그램은 입력 프롬프트전에 키버퍼를 클리어시켜 버퍼에서 대기중인 모든 불필요한 키입력을 제거시켜야 한다. 이 버퍼는 1바이트 ASCII코드나 2바이트 확장코드를 15개까지 저장하는 것이 가능해야 한다. 그러므로 버퍼는 각 키입력에 대해 2바이트 메모리를 마련해 주어야 한다.

즉 1바이트코드의 경우 처음 바이트는 ASCII코드를 유지하고 두번째 바이트는 키의 스캔코드를 유지한다. 확장코드의 경우에는 처음 바이트는 ASCII 0를 유지하고, 두번째 바이트는 일반적으로 키의 스캔코드이지만 일부 키는 시프트키와 결합하여 1코드 이상을 발생시키므로 반드시 두번째 바이트가 스캔코드가 되지는 않는다.

버퍼는 선입선출(FIFO)버퍼로 알려진 원형행렬로 설계되어 있다. 다른 버퍼와 마찬가지로 이 버퍼도 인접 메모리번지의 영역을 사용한다. 그러나 어떤 특정 메모리 위치는 버퍼라인의 앞쪽에 있지 않고 버퍼내에서 자주 사용되는 문자스트링의 HEAD와 TAIL 포인터를 항상 지시한다.

또한 새로운 키입력은 TAIL 다음 장소(메모리에서 상위어드레스를 향한 장소)에 저장되고, TAIL포인터는 이 새로운 저장에 따라 조정되는데, 한번 버퍼의 최상위 메모리가 채워지면 다음에 입력되는 새로운 문자는 버퍼의 최하위 메모리로 되돌아와서 저장된다. 이와 같이 버퍼내 스트링의 HEAD가 TALE보다 더 상위 메모리에 존재하는 경우가 가끔 있다.

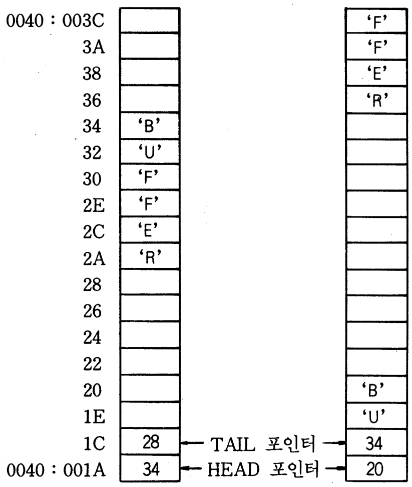
그리고 버퍼가 완전히 채워지면 새로 추가되는 문자는 무시되고, 키보드 인터럽트는 이를 스피커를 통해 알려준다.

버퍼의 데이터 베열을 나타낸 그림 2를 보면, HEAD포인터는 첫번째 키입력을 지시하는 반면에 TAIL 포인터는 마지막 키입력 다음 장소를 지시한다. 두 포인터가 일치하는 경우에는 버퍼가 비어 있음을 의미한다. 15개의 키입력을 갖추기 위해서는 16번째에는 더미 포지션(Dummy Position)이 필요하고 이 더미 포지션 2바이트는 항상 캐리지 리턴(Carriage Return : ASCII 13)과 Enter의 스캔코드(28)를 담고 있다. 이 더미 포지션은 키입력 스트링의 HEAD 바로 전에 있다.

이 32바이트 버퍼는 0040:001E 메모리위치에서 시작하고 HEAD포인터와 TAIL포인터는 각각 0040:001A 및 0040:001C 메모리위치에서 시작된다. 포인터는 2바이트 길이이지만 최하위 최소 유효바이트만이 사용된다. 포인터의 값은 BIOS 데이터영역내의 위치에 따라 30에서 60사이까지 변화한다. 이 버퍼를 클리어시키려면 간단히 0040:001A의 값과 일치하도록 세트시키면 된다.

또 프로그램에 의해 문자를 버퍼에 삽입시키면서 캐리지 리턴으로 스트링을 끝냄에 따라 버퍼포인터를 조정할 수 있는데, 만약 이 기법을 한 프로그램에서 빠져나오기(Exit) 바로 전에 실행시킨다면 DOS상에서도 문자를 읽고 다음 프로그램도 자동적으로 로드될 수 있게 된다.

〔그림 2〕 키보드 버퍼의 데이터 배열



●베이직의 경우

베이직에서 버퍼포인터의 값을 변경 및 페치시키기 위해서는 PEEK와 POKE를 사용한다.

```
100 DEF SEG=&H40              '최하위 메모리번지로
                              '세그먼트를 설정
110 POKE &H1C,PEEK(&H1A)      '포인터를 같게
```

그러나 이 방법은 신뢰성이 없다. 일부 응용에서는 메모리의 다른 부분에 버퍼를 생성시킬 수 있으며, 또한 키보드 인터럽트가 라인 110의 중앙으로 끼어 들어가면서 TAIL포인터를 바꾸어놓을 가능성도 조금은 있다. 이러한 이유 때문에 버퍼포인터를 단독으로 남겨두는 것이 좋다. 대신 널(Null : Zero)이 될 때까지 버퍼의 내용을 읽어낸다.

```
100 IF INKEY$<>"" THEN 100 '만일 0이 아니면
                           '다시 키입력한다'
```

●어셈블리어의 경우

INT 21H의 기능 C는 DOS 키보드 입력기능 1, 6, 7, 8, 및 A중 어느 한 기능을 수행하지만 키보드 버퍼를 먼저 클리어시킨다. 입력 기능번호를 AL에 놓으면 간단히 키보드 버퍼는 클리어된다(여기서 기능번호 AL은 1임).

```
;---키입력 대기 전에 버퍼를 클리어:
      MOV   AH,0CH     ;DOS기능을 0CH로 선택
      MOV   AL,1       ;키입력 기능선택
      INT   21H        ;버퍼를 클리어하고
                       ;키입력동안 대기
```

베이직의 예에서와 같이 TAIL포인터 값과 HEAD포인터값을 같게 한다. 키보드 인터럽트에 의한 간섭을 배제하기 위해서 이 작업을 할 동안에는 인터럽트를 불가능하게 한다.

```
;---HEAD와 TAIL포인터를 같게:
      CLI                  ;인터럽트 불가능
      SUB   AX,AX          ;AX=0로 함
      MOV   ES,AX          ;ES를 최하위 메모리번지
                           ;로 설정
      MOV   AL,ES:[41AH]   ;HEAD포인터를 AL로 옮김
      MOV   ES:[41CH],AL   ;AL을 TAIL포인터로 옮김
      STI                  ;인터럽트 다시 가능
```

키입력을 위한 버퍼체크

실제로 버퍼에서 문자를 제거시키지 않고도 키보드 입력이 가능한지를 체크할 수 있다. 버퍼는 일반적으로 버퍼내 문자열의 처음과 끝을 가리키는 두 포인터를 사용한다. 앞에서 언급한 바와 같이 두포인터가 일치하면 버퍼는 비어있다는 뜻이다. 따라서 단지 0040 : 001A와 0040 : 001C 메모리위치가 같은지를 비교하면 된다.(그러나 버퍼는 원형행렬로 구성되어 있고 행렬시작은 계속 위치가 바뀌기 때문에 단지 행렬시작 문자만을 체크할 수는 없다)

●베이직의 경우

2바이트를 읽고 비교하기 위해서 PEEK를 사용한다.

```
100 DEF SEG=&H40           '0으로 메모리 세그먼트를
                           '설정
110 IF PEEK(&H1A)<>PEEK(&H1C) THEN...
```

● 어셈블리어의 경우

기능 B 또는 INT 21H는 키보드 버퍼가 1이상의 문자를 갖고 있을 때에는 AL레지스터에 FFH를 리턴시키고 버퍼가 비어 있을 때는 00을 리턴시킨다.

```
;---문자가 버퍼에 있는지 체크:
      MOV   AH,0BH           ;기능번호 전송
      INT   21H              ;인터럽트 21 호출
      CMP   AL,0FFH          ;FF와 비교
      JE    GET_KEYSTROKE    ;존재하면 점프
```

BIOS의 INT 16H 기능 1도 같은 기능을 수행하며 이 기능에 추가로 어떤 문자인지를 보여준다. 제로 플래그(ZF)는 버퍼가 비어있을 때는 1로, 문자가 대기중일 때에는 0으로 세트된다. 0으로 세트될 경우에, 버퍼 HEAD에서의 문자의 복사는 버퍼에서 이를 제거하지 않고 AX에 놓는다. AL은 1바이트 ASCII문자의 경우 문자코드로, 확장코드의 경우에는 ASCII 0으로 되며 이 코드번호인 경우는 AH에서 나타난다.

```
;---문자가 있는지 찾는다:
      MOV   AH,1            ;기능번호
      INT   16H             ;버퍼에 문자가 있는지
                            ;체크
      JZ    NO_CHARACTER    ;제로 플래그가 1이면
                            ;점프
;---문자가 있으면 어떤 것인지 알아본다:
      CMP   AL,0            ;확장코드인지 체크
      JE    EXTENDED_CODE   ;맞으면 확장코드루틴
                            ;으로 간다. 아니면 AL
                            ;에서 문자를 가져온다
```

베이직 예에서와 같이 두 버퍼 포인터를 비교해 본다.

```
;---HEAD와 TAIL포인터를 비교:
      MOV   AX,0            ;특별한 세그먼트 사용
      MOV   ES,AX           ;세그먼트를 0으로 설정
      MOV   AL,ES:[41AH]    ;한 포인터를 전송
      MOV   AH,ES:[41CH]    ;또 다른 포인터를 전송
      CMP   AH,AL           ;포인터를 비교
      JNE   GET_KEYSTROKE;같지 않으면 입력 루틴
                            ;으로 점프
```

입력된 내용을 화면에 나타내지 않을려면

일반적으로 입력된 키입력은 스크린으로 되돌려져 입력되는 내용을 보여준다. 그러나 어떤 경우에는 입력내용을 화면으로 나타내고 싶지 않을 때가 있다.

일 예로서 메뉴선택을 필요로 하는 프로그래밍 작업에서는 해당 메뉴를 선택한 키입력을 굳이 화면에 나타낼 필요가 없다. 그리고 어떤 경우에는 입력 문자가 스크린상에 보내지기 전에 에러체크가 필요할 때도 있다. 특히 확장코드를 받아들이는 일부 프로그램에서는 이 점에 대해서 세심한 주의가 필요한데, 그 이유는 이 확장코드의 첫 바이트(ASCII 0)가 디스플레이되면서

문자사이에 공간을 남겨두기 때문이다.

●베이직의 경우

베이직의 INKEY$ 기능은 키입력의 내용을 화면에 나타내지 않으면서 입력할 때 사용된다. 이는 ASCII문자일 때는 1바이트 길이의 스트링을, 확장문자일 때는 2바이트 길이의 스트링을 리턴시킨다. INKEY$는 문자가 들어올때까지 계속하여 순환되는 루프내에 입력되지 않은 키입력을 위해 대기하지는 않는다. 이 루프는 INKEY$를 불러내어 INKEY$가 리턴시킨 스트링을 하나의 변수(여기서는 C$)로 지정하여 루프기능을 수행한다. 키입력을 받지 못할 경우에는 INKEY$는 사이에 아무 것도 없는 두 따옴표를("")로 표시되는 0문자 길이의 스트링인 널스트링을 리턴시킨다. INKEY$가 ""를 리턴시키는한 이 루프는 계속 반복된다. 즉

```
100C$=INKEY$:IFC$="" THEN 100.
```

이다.

다음 예는 키입력에 의한 메뉴선택이고, 각 메뉴선택에 의해 프로그램을 특정 서브루틴으로 보낸다고 가정한 것이다. 이 선택은 A, B, C 등을 쳐서(1바이트 ASCII 코드결과를 냄)하거나 또는 Alt-A, Alt-B, Alt-C 등을 쳐서(2바이트 확장코드 결과를 냄) 이루어진다. 이 두 경우가 어떻게 다른지는 LEN기능을 사용하여 스트링이 1문자 길이인지 2문자 길이인지를 체크해 보면 알 수 있다.

만약 1바이트 ASCII코드이면 일련의 IF...THEN 명령문이 즉시 키입력의 동일함을 시험하기 시작하고 이 프로그램을 적절한 서브루틴으로 보낸다. 한편 2바이트 코드의 경우에 콘트롤은 단독 루틴으로 이동된다. RIGHT$기능은 왼쪽 문자를 제거시키는데 이 왼쪽 문자는 확장코드를 0과 같게 하는 것외에는 없다. 이를 제거후 ASC기능으로 문자를 스트링형에서 수치형(Numeric Form)으로 변환시킨다.

최종적으로 제2의 IF...THEN 등의 명령어들은 Alt-A, Alt-B 등에 해당되는 번호에 대한 결과를 체크하게 된다(리스트 1 참조).

라인 120(이하 라인 포함)에서 "a" 및 "A" 대신에 ASCII코드의 수치값을 사용할 수도 있다.

```
120 IF C=97 OR C=65 THEN GOSUB 1100
```

물론 처음의 C$는 라인 510에서와 같이 정수형으로 변환될 필요가 있을 수 있다. 그리고 명령문의 길이가 긴 프로그램에서 C를 변환시켜서 변환된 C가 한 문자의 소문자나 대문자형태중 하나를 항상 나타내게 함으로서 공간을 절약할 수도 있다.

먼저 에러체크를 하여 C$의 ASCII 값이 적정 범위내에 있는지를 확인한다. 그리고 난후 그 수가 91보다 낮은 수인가를 알아 보는데, 이 경우 대문자이면서, 91보다 낮은 수가 아닌 경우에는 그대로 두면 된다. 이렇게 변환시킨 후에는 "IF C=97 THEN..."과 같은 짧은 명령문으로도 충분하게 될 것이다. 리스트 2는 그 코드에 대한 예이다.

●어셈블리어의 경우

INT 21H의 기능 7과 8은 키보드 버퍼내에 아무 것도 없으면 문자를 위해 대기하고 있다가 문자가 들어와도 스크린상에 나타내지 않는다. 기능 8은 Ctrl-Break을 감지(그리고 Ctrl-Break 루틴을 초기화)하는 반면에 기능 7은 그렇지 않는다. 두 경우 모두 문자는 AL로 리된다. AL이 ASCII 0을 갖고 있으면 확장코드가 들어 온 것이다.

인터럽트를 반복시키면 코드의 두번째 바이트가 AL에 나타난다(리스트 3 참조).

BIOS는 DOS기능과 같이 AH에 0을 놓고 INT 16H를 호출하는 기능을 갖고 있다. 이 기능은 문자를 대기하고 AL에 리턴시키는 일을 한다. 이 경우에 확장코드는 인터럽트를 단 한번만 호출하면 된다.

〔리스트 1〕

```
100 C$=INKEY$:IF C$="" THEN 100           '키입력동안 대기
110 IF LEN(C$)=2 THEN 500                 '확장코드면 점프
120 IF C$="a" OR C$="A" THEN GOSUB 1100 'A에 대한 메뉴선택
130 IF C$="b" OR C$="B" THEN GOSUB 1200 'B에 대한 메뉴선택
140 IF C$="c" OR C$="C" THEN GOSUB 1300 'C에 대한 메뉴선택
.
.
500 C$=RIGHT$(C$,1)                     '확장코드의 2번째 바이트를 얻음
510 C=ASC(C$)                           '수치값으로 변환
520 IF C=30 THEN GOSUB 2100             'Alt-A에 대한 메뉴선택
530 IF C=48 THEN GOSUB 2200             'Alt-B에 대한 메뉴선택
540 IF C=46 THEN GOSUB 2300             'Alt-C에 대한 메뉴선택
```

〔리스트 2〕

```
500 C=ASC(C$)                                          '문자의 ASCII번호를 설정
510 IF NOT((C>64 AND C<91) OR (C<96 AND C<123)) THEN... '...범위를 넘은 것은 무시
520 IF C<91 THEN C=C+32                                '91보다 작은 대문자의 값에 32를 더함
530 IF C=97 THEN...                                    '...그 다음 그 문자를 테스트
```

만일 AL에 0이 나타나면 확장코드 번호는 AH에 존재한다. 여기서 Ctrl-Break는 검출되지 않는다(리스트4 참조).

키입력을 대기하고 이를 화면으로 나타내려면

텍스트나 데이터 엔트리(Entry)의 키입력은 일반적으로 스크린상에 나타난다. 스크린상에 나타낼 때 캐리지 리턴이나 백스페이스와 같은 문자들은 해당문자의 ASCII 기호를 디스플레이 시키는 것보다는 오히려 커서를 적당히 이동시키는 것으로 해석된다. 스크린상에서 표현은 커서가 일반적으로 세트되는 곳에서 시작되고, 텍스트는 그 줄의 마지막 컬럼에서 자동적으로 다음 라인으로 이어지게 된다. 그리고 이렇게 이어진 문자는 당연히 비디오 버퍼상의 다음 위치에 놓인 것이 됨으로 특별한 코딩이 필요치 않다. 그리고 이 버퍼는 필수적으로 스크린상의 25 라인을 한번에 이을 수 있어야 한다.

●베이직의 경우

베이직에서는 INKEY$를 사용하는 키입력을 인터셉트(Intercept)하고 난 후 다음에 다시 인터셉트하기 위한 리턴을 하기 전에 이를 프린트한다. 이때 PRINT명령문을 사용하거나 또는 POKE문을 이용하여 메모리 맵핑기법으로 키입력을 직접 비디오버퍼에 넣는다. (이 버퍼는 단색 어댑터에서는 메모리 세그먼트 &HB000에서 시작하고, 컬러 어댑터는 &HB800에서 시작된다.) 그런데 PRINT를 사용할 때에는 명령문을 꼭 세미콜론으로 끝내야 한다. 그렇지 않을 경우에는 캐리지 리턴이 자동적으로 이루어진다.

다음은 이에 대한 예이다. 여기서 비문자 키입력은 정렬(Sort Out)시키지 않았다. 변수 KEYSTROKES$는 키입력에 의해 입력된 내용을 데이터 스트링에 모은다(리스트 5 참조).

〔리스트 3〕

```
;---키입력 얻음:
               MOV  AH,7          ;기능번호를 설정
               INT  21H           ;문자에 대한 대기 실행
               CMP  AL,0          ;확장코드인지 알아본다
               JE   EXTENDED_CODE ;그러면 확장코드루틴으로 간다.
               .                  ;아니면 AL로 문자를 옮김
               .
;---확장코드 루틴:
EXTENDED_CODE: INT  21H           ;현재 확장코드번호는 AL에 있는 것으로 실행
               CMP  AL,75         ;커서가 왼쪽으로 가는지 체크
               JNE  C_R           ;아니면 다음 가능한 것을 체크
               JMP  CURSOR_LEFT   ;그러면 루틴으로 점프
C_R            CMP  AL,77         ;등
```

〔리스트 4〕

```
;---키입력을 얻음:
               MOV  AH,0          ;인터셉트의 키입력에 기능번호
               INT  16H           ;키입력을 실행
               CMP  AL,0          ;확장코드인지 체크
               JE   EXTENDED_CODE ;그러면 특별 루틴으로 점프
                                  ;아니면 AL로 ASCII문자를 가져가 비교
;---확장코드 루틴:
EXTENDED_CODE: CMP  AH,75         ;AH로 확장코드를 가져가 비교
```

〔리스트 5〕

```
100 ''method using PRINT: 110 LOCATE 10;40 '40행,10열 위치에 커서를 설정
120 KEYSTROKES$=""                         '입력 스트링을 유지할 수 있도록 변수를 클리어
130 C$=INKEY$:IF C$="" THEN 130            '키입력을 얻음
140 KEYSTROKES$=KEYSTROKES$+C$             '스트링변수에 키입력을 더함
150 PRINT C$;                              '문자를 프린트
160 GOTO 130                               '다음 문자를 실행

100 '''method using POKE(monochrome adaptor):
110 DEF SEG=&HB000                         '스타트버퍼로 오프셋 세그먼트를 설정
120 POINTER=1678                           '10의 위치는 40=(2*((10*80)+40))-2
130 KEYSTROKES$=""                         '입력스트링을 유지할 수 있도록 변수를 클리어
140 C$=INKEY$:IF C$="" THEN 140            '키입력을 얻음
150 KEYSTROKES$=KEYSTROKES$+C$             '스트링변수에 키스트록을 더함
160 POKE POINTER,ASC(C$)                   '버퍼내에 문자의 ASCII번호를 찾음
170 POINTER=POINTER+2                      '2만큼 포인터를 올림(속성바이트 스킵)
180 GOTO 140                               '다음 문자를 실행
```

●어셈블리어의 경우

INT 21H의 기능 1은 키보드 버퍼내에 문자가 없는 경우에는 대기하며, 이 문자를 스크린상의 현재 커서위치에

〔리스트 6〕

```
;---키입력을 얻음:
            MOV   AH,1            ;기능번호 설정
            INT   21H             ;문자를 위한 대기 실행
            CMP   AL,0            ;확장코드인지 비교
            JE    EXTENDED_CODE   ;그러면 특별한 루틴으로 점프 아니면 AL로 부터 ASCII문자를 가져옴
            .
            .
;---확장된 코드 루틴:
            INT   21H             ;AL로 코드번호를 가져옴
            CMP   AL,77           ;커서 오른쪽으로 이동하는지 체크
            JNE   C_R             ;그렇지 않으면 다음 가능한 것을 체크
            JMP   CURSOR_RIGHT    ;그러면 루틴으로 점프
  C_R:      CMP   AL,75
```

나타낸다. Ctrl-Break은 Ctrl-Break루틴(프로그램 가능)이 실행되도록 인터셉트된다. 문자는 AL에 리턴되고 확장코드의 경우, AL에 가져오기 위해 반복한다(리스트 6 참조).

이 기능은 "Esc(Escape)"키를 완전히 무시하고, 탭(Tap) 키입력을 정상적으로 해석한다. 백스페이스키는 커서를 1스페이스 뒤로 이동하게 하지만 그 위치에 있던 문자는 지워지지 않는다. 엔터(Esc : 리턴)키는 커서를 현재 라인의 시작점으로 이동시킨다(자동 라인피드(Feed)기능은 없다). PC

디렉토리에 관한 모든 기능을 가진 별난 소프트웨어
DX.COM

김기준/아주대 전자과

이번에 소개할 공개소프트웨어 디스켓은 DIREXT, PCOUTLN, SK152의 3개 디렉토리로 구성되어 있다. PCOUT-LN속에는 계층적 구조와 워드프로세서 기능을 통하여 각종 환경에 적합한 프린팅화일을 만들어 주는 PCO.EXE와 관련 유틸리티 프로그램들이 들어있고 SK152에는 독자들도 즐겨 쓰고 있을 사이드 킥(SideKick) 프로그램(버전1.52)이 들어있는데 이것은 익히 잘 알고 있듯이 노트패드를 이용한 화일작성 및 캘린더 참조, 계산기기능 등 여러가지 기능들이 들어 있으며 이들은 언제라도 손쉽게 불러내어 이용할 수 있다(화면 1 참조). (지금은 신버전인 사이드킥 플러스를 주로 쓰나 다른 작업을 하면서 간편하고 빠르게 이들 기능을 이용할 때는 이러한 구버전이 도리어 편리할 때도 있을 것이다.)

표 1은 이 디스켓의 화일들을 설명한 것이며 지면관계상 여기서는 DIREXT 안의 DX 유틸리티를 중심으로 설명할까 한다. 사이드킥은 시중에 매뉴얼이 많이 나와 있으므로 이들 자료를 참조하기 바라고 PCOUTLN은 2개의 README 화일과 실행도중에 나오는 잘 정리된 메뉴를 참조하면 쉽게 배울 수 있으리라 본다(화면 2 참조).

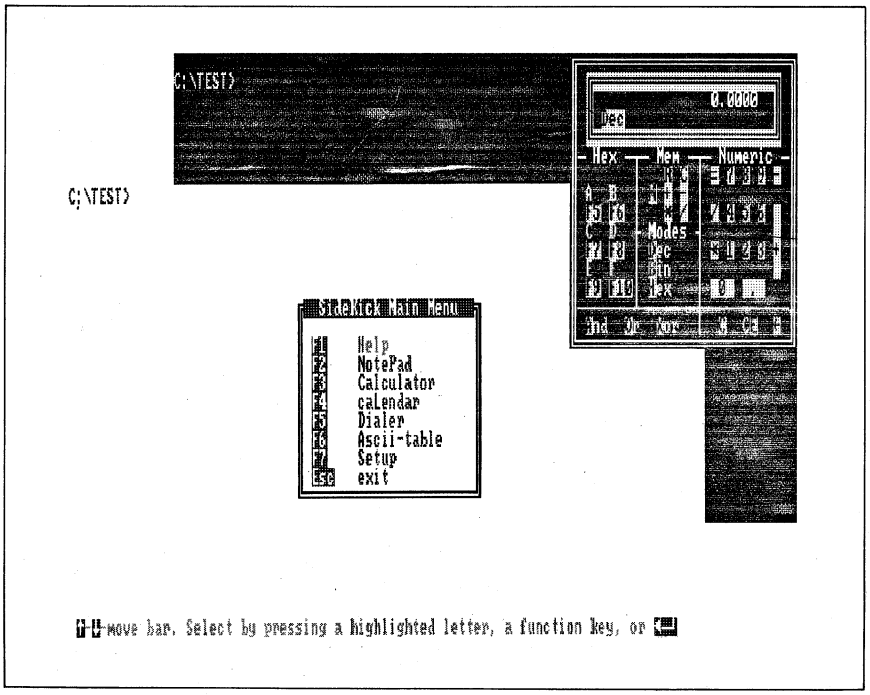
소개

DX는 Directory eXtended의 약자로써 ECONO-SOFT라는 곳에서 만든 공개 소프트웨어, 즉 다시 말하자면 쉐어웨어로서 오로지 디스크와 화일의 관리를 위해 만들어진 유틸리티이다.

여지껏 우리 유저들은 어떤 불편함에 앞서서 아쉬움때문에 DOS의 내부명령어인 DIR의 단순한 옵션들 '/W'(화일을 5개씩 표시), '/P'(페이지 단위로 디렉토리의 내용표시)와 와일드카드 문자인 *,?에 만족해야 했다. 하지만 좀더 많은 기능을 갖는 유틸리티들이 사용되어 온 것이 사실이다. 예를 들면 컴퓨터를 써본 사람이라면 모르는 사람이 없는 노턴 유틸리티, PC-tools 등이 그것인데 만약 일정기간 동안에 만들어진 화일을 찾는다고 할 때, PCtools는 훌륭한 기능을 발휘할 수 있지만 이 경우 일일이 DIR을 입력해서 모든 디렉토리를 확인한다면 여러 단계의 손을 거쳐야 할 것이다.

그런데 DX가 이들보다 더 훌륭한 기능을 발휘한다면 어떨까? 한 예로 임의의 어떤 화일을 찾을려고 할 때 노턴 유틸리티중 FF.EXE라는 File Find 유틸리티 이외에 퀵도스(QuickDos), PCtools 등에서도 막강한 찾기 기능을 갖추고 있는데 이들 유틸리티에서 제공하는 기능은 겨우 어떤 패턴을 주고 그것에 일치가 되는 것만을 찾는 단순한 기능에 불과하다. 그러나 DX에서는 그러한 기본 기능에다 어떤 예쁜(?)

〔화면 1〕 사이드킥의 실행화면



수식어를 덧붙여 줌으로써 더욱 더 세련된 기능을 얻을 수 있다. 가령 어떤 화일을 찾는데 처음 시작되는 철자와 어느 기간사이에 저장을 시켰다는 것 밖에 모를 경우도 DX는 멋있게 화일을 찾아낼 수가 있는 것이다. 이 글을 좀 더 읽다 보면 이 정도 기능은 DX에서는 새발의 피(?)에 불과하다고 말할 수 있다. 그 이유는 너무나 강력하고, 세밀하고, 멋있는 기능들이 이 자그마한 화일 곳곳에 숨어 있기 때문이다. 그러나 옥에 티라고나 할까? 한가지 단점이 있다면 처음 써보는 유저의 입장에서는 너무 많은 기능 때문에 오히려 사용하기가 난해할 수 있다는 점이다. 그러나 필자의 생각으로는 일단 숙련이 된다면 이것처럼 멋있는 기능을 가진 유틸리티도 찾아보기 힘들 것이라는 생각이 든다.

그리고 DX는 터보 파스칼 3.0으로 작성되었고 내장 명령어가 아닌 하나의 화일로서 실행되기 때문에 읽어 들이는데 시간이 조금 더 걸린다는 단점이 있지만 이 유틸리티를 만든 곳에서는 장담하기를 앞으로의 새로운 버전에서는 지금의 속도에 비해 50%나 빨라질 것이라고 하니 일단은 아쉬운데로 이 막강한 기능을 가진 DX라는 괴물 유틸리티를 써보도록 하자.

다음은 DX의 실행환경을 보인 것이다.

- IBM PC, IBM PC XT / AT, IBM PCjr, 3270 PC, XT / 370, COMPAQ, COMPAQ Plus와 호환 기종
- 사용할 수 있는 메모리는 120K바이트 이상(램상주 프로그램이 로드되어 있을 경우 여분의 사용할 수 있는 메모리는 120K바이트 이상이어야 한다.)
- 컬러 또는 모노크롬 모니터(물론 허큘리스카드 포함)

MS / PC-DOS 2.x 또는 3.x 이상

〔표 1〕 주요 화일의 내용

디렉토리	화 일 명	기 능
DIREXT	DX-HELP.COM	화면 메뉴를 통해서 DX의 기능을 얻는다.
	DX.COM	디렉토리 관리기능을 갖는 유틸리티
PCOUTLN	BEEP.COM	'beep' 음을 낸다. 배치화일에서 에러 발생이나 처리 종료를 나타내는데 이용할 수 있다.
	GLOBAL.COM	DOS기능을 전 디렉토리 범위에서 수행시킨다.
	CPY.COM	일련의 화일들을 플로피디스크로 백업한다.
	DIRS.COM	일련의 화일들에 의해 채워진 디스크 용량을 보여준다.
	MOVE.COM	도중에 복사작업을 거치지 않고 한 디렉토리에서 다른 곳으로 화일을 옮긴다.
	MEM.COM	CHKDSK를 거치지 않고 남은 메모리 용량을 보여준다.
	PUSHDIR.COM	배치화일로부터 현재의 디렉토리를 기억한다.
	POPDIR.COM	이전의 PUSHDIR로부터 커런트 디렉토리를 복원한다.
	PCO.EXE	PC-OUTLINE를 실행하는 화일
	DOC.PCO	PC-OUTLINE 화일에 대한 도큐먼트 화일
	EPSON.CFG	엡슨 프린터 드라이버
	BLANK.CFG	모든 프린터 코드를 제거해 버리는 더미 프린터 드라이버
	IBM.CFG	IBM 프린터 드라이버
	PCPRINT.COM	프린터 드라이버를 바꿔주는 프로그램
	KEYSET.COM	PC-OUTLN의 키를 커스텀화시키는 프로그램
SK152	SK.COM	사이드킥(버전 1.52) 실행 화일
	SKINST.COM	사이드킥 인스톨 프로그램

특징

다음은 DX가 갖는 특징을 요약해 본 것이다.

① DOS의 DIR명령과 같이 모든 버전의 DOS에서 지원된다.

② 모든 경로(Path)와 하드디스크 드라이브를 지원한다.

③ 매우 강력하고 유동적인 화일 매칭 구조를 갖고 있어서 문자 스트링 안에 여러개의 와일드카드 문자를 허용한다.

④ 여러 경우에 있어서 특별한 환경을 제공해 준다. 예를 들면 "*.BAS+*.PAS; *T"는 *.BAS와 *.PAS 중에서 T로 끝나지 않는 패턴을 찾는 것을 말한다.

⑤ DX는 모든 디렉토리와 디렉토리내의 모든 화일 리스트, 그리고 다른 디스크에 있는 화일이 일치되는지의 여부를 테스트 해준다.

⑥ 화일의 속성을 조정할 수 있다.(DX에서는 아무 속성이 없는 'Normal'을 제공한다.)

⑦ 어느 곳에든지 Hidden File, System File을 볼 수 있다.

⑧ '.COM', '.EXE', '.BAT' 같은 실행화일 만을 찾아서 출력할 수 있다.

⑨ DX는 디렉토리안에 존재하는 '.', '..'를 표시하지는 않으나 보기를 원한다면 보이게 할 수 있다.

⑩ 배치화일 오퍼레이션 기능이 있다.

⑪ DX는 화일이름, 확장자, 화일크기, 날짜, 시간 등 여러가지에 의해서 소팅을 시킬 수 있다.

⑫ 화일의 속성을 변경시키거나 모두 지울때 확인 또는 확인이 없이 화일을 지울 수 있다.

⑬ 데이터 상태를 체크하여 화일의 동질성을 확인하는 체크섬(CheckSum) 이 가능하다.

⑭ 다음과 같이 확정된 출력 형식을 갖는다.

- 컬러 표현
- 프린터 포매팅
- 대문자, 소문자 또는 혼용(첫자가 대문자로)으로 출력

• 12, 24시간 형식을 지정하여 출력

⑤ DX는 자주 이용되는 스위치 등을 간략화하여 사용할 수 있다.

⑥ 특별한 경우에 사용되는 유틸리티는 제외되겠지만, 기능이 매우 많으므로 디스크를 차지하는 비율을 줄일 수 있다.

이상으로 간략하게나마 앞으로 설명할 DX의 여러가지의 특징을 살펴보았는데 이제 위의 기능들을 직접 구현시킬 수 있도록 하는 여러가지의 옵션 스위치에 대하여 알아 보겠다.

사용법

먼저 DX를 사용함에 앞서서 사용문법 및 기본 규칙을 보면 다음과 같이 나타낼 수 있다.

```
DX [ /switch {/switch} ] [ filespec ]
                    [/switch {/switch}]
```

• 화일 스펙을 위한 문법

```
[d:] [path] [ [ ; ] pattern
          { + pattern ¦ ; pattern } ]
```

• 스위치 구성을 위한 문법

```
/name [+¦-¦:[-] value [:value]¦=
                    [-]value[:value]]
```

• 위의 옵션이 생략되어 있다면 디폴트는 *.*이 된다.

• 스위치의 시작은 슬래시('/')이며 패스는 역슬래시('\')로 시작된다.

• '='라는 기호는 ':'로 대치될 수 있다.

• DX / sw는 스위치 ON을, DX / sw-는 스위치의 OFF를 의미한다.

• 각 스위치는 약어로써 표시될 수 있다.

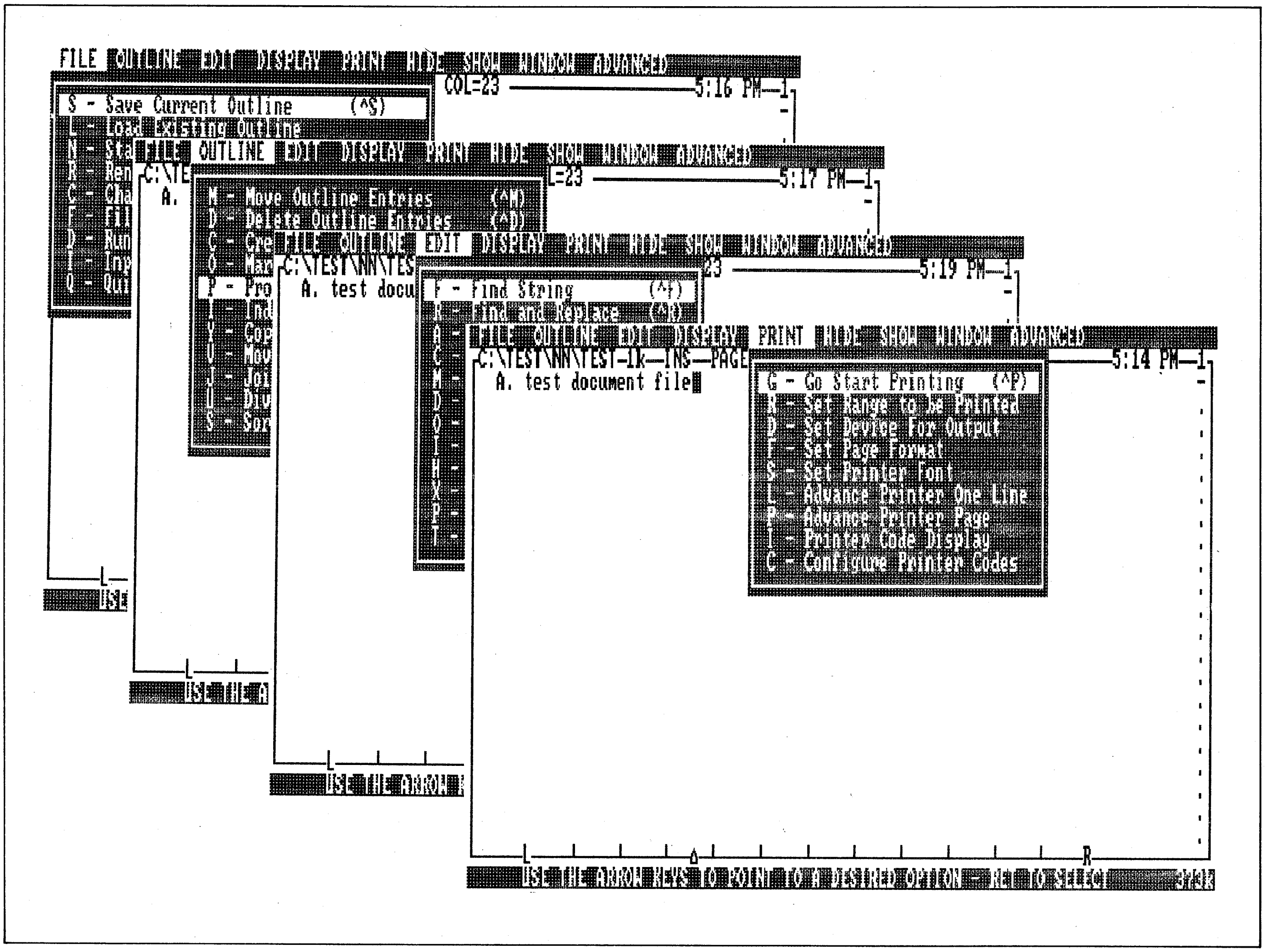
예를 들면 Pause의 경우 / p, / pa, / pau, / pause 모두 같은 의미가 된다. 그러나 약어가 중복되는 경우도 있으므로 주의를 요한다.(/ Single과 /Si의 경우)

이제 기본적으로 알아야 할것을 알았으므로 본론으로 돌아가 스위치들의 종류를 살펴 보도록 하자.

(1) 포매팅 옵션

• / Pause : DOS의 내부 명령어인 dir

〔화면 2〕 PCO의 OUTLINE 메뉴선택화면 (각 메뉴에 따라서 풍부한 선택 윈도우가 준비되어 있다.)



/p와 거의 같은 기능을 갖는다. 모니터에 화일들이 꽉 차게되면 키의 입력을 기다리며, 키가 눌러지면 다음의 화일이 출력된다. DX.COM화일이 들어있는 디렉토리에서 다음을 실행시켜 보자.

```
DX/p
DX C:\DOS/P
DX/pause autoexec.bat
```

화면 3은 c:\utility라는 디렉토리중 ＊.com과 ＊.exe라는 화일을 찾고 페이지 단위로 출력하는 예이다. 여기서 보면 각 페이지마다 'hit any key to continue...'라는 메시지를 볼 수 있을 것이다. 그리고 DX는 c:\utility 디렉토리 안에 있는 또다른 디렉토리에서도 목적 화일을 찾아주고 있다.

- /Wide : 이 옵션은 DOS에서의 dir/w와 같은 기능을 갖는다. 즉 한줄에 5개의 화일 또는 디렉토리가 출력된다.

```
DX/w
DX/wide
```

- /SINgle : 이 옵션은 DOS에서의 dir과 같은 기능을 갖는다. 한줄에 1개의 화일이름, 크기, 시간 등이 출력된다. 그런데 여기서 우리들은 DX/SI와 혼동을 피해야하는데 SI는 'System Information'이라는 옵션으로 인식을 해버리므로 주의를 해야한다.

```
DX/SIN
DX *.pas/sin
```

- /Total : 전체적인 통계만이 출력된다. 화면 4는 이 옵션을 사용하여 출력한 결과이다. 여기서 우리는 2개의 한정자인 'S'와 'G'를 볼 수 있는데 출력 형태가 서로 다름을 볼 수 있다(화면 5 참조).

화면 5를 보면 (a)에서는 'S'옵션을 이용해서 출력을 한정시켜 주므로 ＊.exe라는 화일이 현재 디렉토리 안에서 차지하는 크기와 갯수만을 보이고 있

〔화면 3〕 특정 형태의 화일을 페이지 단위로 출력시킨 예

```
C:\>cls

C:\>dx c:\utility\.com+.exe/p/ff

Directory eXtended (TM)  v1.07;  C/N 00605.          (Type DX/INFO for info.)
Copyright (c) 1985, 1986 Robert K.Blaine/ECONO-SOFT.

Volume in drive C is [KIM GI JOON]
Directory of [C:\Utility\.Com+.Exe]

Ep.Exe          2608 a   88dec08 18:58 | E2.Exe          73472 a   85aug16 12:00
Pz.Com         16640 a   88sep19 10:43 |

     92,720 bytes in 3 file(s) in this directory.

Directory of [C:\Utility\Crosstal\.Com+.Exe]

Command.Com    23210 a   85mar07 13:43 | Copy-Ii.Exe     17792 a   83dec19  9:07
Diskcopy.Com    2576 a   83oct20 12:00 | Edlin.Com        4608 a   83oct20 12:00
Hp.Exe         10088 a   80jan01  2:53 | Print.Com        4608 a   83oct20 12:00
Ws.Com         21376 a   80jan01  0:10 | Xtalk.Exe       56960 a   83jun27 15:36

    141,218 bytes in 8 file(s) in this directory.

Directory of [C:\Utility\Crosstal\Rs232c\.Com+.Exe]

Basic.Com      16256 a   83oct20 12:00 | Command.Com     17792 a   83oct20 12:00
Fileconv.Exe   39040 a   82aug27  0:07 | Rs232int.Exe     5504 a   82aug27  0:07
hit any key to continue ... 

Gwbasic.Exe    67392 a   80jan01  2:25 |

    145,984 bytes in 5 file(s) in this directory.

Directory of [C:\Utility\Dbase\Handb\.Com+.Exe]

Hgc.Com          507     80jan01  0:31 | Command.Com     17792     80jan01  0:05
Dbaseovr.Com   37888     80jan01  0:10 | Dsort.Com       15616     80jan01  0:11
Edli[illegible]          [illegible]  0:11 | Handb.Com       31616     80jan01 [illegible]
[illegible]Exe       10088     80jan01 [illegible]                                  0:11

Command.Com    17792 a   83oct20 12:00 | Imageact.Exe   216352 a   80jan01  0:33
No2.Com         3200 a   84dec11 12:00 | Prolok.Exe      52816 a   84nov14 13:25

    290,160 bytes in 4 file(s) in this directory.

  2,391,537 bytes in 75 file(s).         2,461,696 bytes allocated (3% slack).
    202,752 bytes free (1%); 13 dir(s).     2,048 byte minimum allocation.
 21,309,440 bytes total on: Hard Disk.           No attributes hidden.

C:\>
```

고 (b)에서는 'G'옵션을 이용해서 (a)에서와는 달리 전체의 통계를 보이고 있다.

• /HEader : 이 옵션을 이용하면 매번 출력시마다 화면 윗부분에 나타나는 머리글을 선택적으로 출력할 수 있도록 해주는 옵션이다. 다음과 같이 입력한다면

```
C:\>dx/p/he:2
```

맨 끝에 :2라는 한정자가 주어져 있기 때문에 2줄로 된 머리말이 화면 출력시 모두 볼 수 있을 것이다. 이번엔 한정자를 변경시키면서 다음과 같이 실행시켜 보기 바란다.

```
dx/p/he:1
dx/p/he:0
```

':1'일 때는 머리말이 한줄만 보여지고 ':0'인 경우는 아무런 머리말이 없이 화면 출력이 이루어지고 있음을 볼 수 있다. 그리고 한가지 주의할 점이 있다면 DX에는 /HEder의 비슷한 약어인 /Hidden이 있으므로 HEader를 설정하기 위해서는 /HE(/he)와 같이 2자를 지정해 주어야 한다는 것이

• /STats : 이 옵션은 위의 /HEader가 머리말의 출력을 제어하는데 반해 화면 출력의 맨끝에 나오는 화일, 디스크, 속성 등의 정보에 대한 출력을 제어한다. 이것 역시 뒤에 :0~:4까지의 한정자를 붙인다.

4…디폴트로써 모든 것을 출력한다.
3…맨 아랫줄의 전체 디스크 공간, 디스크 종류, 숨겨진 화일을 제외한 모든 것을 출력한다.
2…위의 한정된 것에다 추가하여 현재 사용되어진 용량, 사용 안된 영역의 비율, Minimum Allocation 등이 한정된다.
1…화일명과 디렉토리명만이 출력된다.
0…아무것도 출력되지 않는다.

```
dx c:\utility\vaccine /stat:0
dx c:\utility\simcga/stat:2
```

〔화면 4〕 T(Total)옵션을 지정하여 전체적인 디렉토리 정보를 출력시킨 예

```
C:\MASM>dx/t
Directory eXtended (TM)  v1.07;  C/N 00605.      (Type DX/INFO for info.)
Copyright (c) 1985, 1986 Robert K.Blaine/ECONO-SOFT.

Volume in drive C is [KIM GI JOON]    Directory of [C:\Masm\*.*]

    882,156 bytes in 61 file(s).          960,512 bytes allocated(8% slack)
    526,336 bytes free (2%).                2,048 byte minimum allocation.
 21,309,440 bytes total on: Hard Disk.        No attributes hidden.

C:\MASM>
```

(주석: 현재의 디렉토리 표시 / 전체적인 통계만 출력되고 있다.)

〔화면 5〕 'S'와 'G'옵션 비교

```
C:\MASM>dx *.exe/ff/t:s
Directory eXtended (TM)  v1.07;  C/N 00605.      (Type DX/INFO for info.)
Copyright (c) 1985, 1986 Robert K.Blaine/ECONO-SOFT.

Volume in drive C is [KIM GI JOON]
Directory of [C:\Masm\*.Exe]
    500,036 bytes in 20 file(s) in this directory.

C:\MASM>
```

(주석: S옵션 지정 / *.EXE라는 화일이 현재의 디렉토리 안에서 차지하는 크기와 갯수만 제한해서 출력)

(a) S옵션 지정

```
C:\MASM>dx *.exe/ff/t:g
Directory eXtended (TM)  v1.07;  C/N 00605.      (Type DX/INFO for info.)
Copyright (c) 1985, 1986 Robert K.Blaine/ECONO-SOFT.

Volume in drive C is [KIM GI JOON]
    500,036 bytes in 20 file(s).        522,240 bytes allocated (4% slack)
    526,336 bytes free (2%); 1 dir(s).    2,048 byte minimum allocation.
 21,309,440 bytes total on: Hard Disk.        No attributes hidden.

C:\MASM>
```

(주석: G옵션 지정 / 전체적인 통계 출력)

(b) G옵션 지정

• /CAse : 이 옵션은 화면에 디렉토리, 화일 등을 출력할 때 이 이름들의 출력 양식을 대문자, 소문자, 혼용(맨 앞자만 대문자)으로 선택할 수 있는 옵션이다. 여기에도 각각을 한정해주는 한정사가 붙는데 :U는 대문자 : L는 소문자, :I는 디폴트로서 첫자만 대문자로 나타내고 나머지는 소문자로 나타낸다.

```
dx/ca:u
dx *.pas /CAse:1
```

• 12 : 시간의 형태로 바꾸어 주는 옵션이다. 디폴트는 24시간 형태이다.

```
dx c:\utility\simcga/12
```

●컬러 구성

거의 모든 소프트웨어가 컬러를 지원하고 있는 것처럼 DX도 컬러를 지원한다. 물론 이 얘기는 컬러 모니터를 가지고 있는 유저들에게나 적용되는 말이지만, 지금 현실이 거의 대부분의 사용자가 허큘리스 카드를 사용하고 있는 현실을 감안한다면, 조금은 적합하지 않을지 모르나, 그런데로 여러가지 농도 정도는 표시해 줄 수 있다.

DX가 로드되었을 때 만일, 컬러가 지정되어지지 않으면 DX는 현재 커서의 위치의 색깔로 컬러를 세트해 버린다. 이말의 의미는 어떤 컬러라 하더라도 DOS상태에서 지원될 수 있다는 말이 된다. 만약 어떤 특정한 컬러를 원한다면, 3개의 스위치를 지정해 줌으로써 원하는 컬러를 얻을 수 있다. 먼저 각각의 스위치는 표 2의 컬러 데이터를 기초로 하고 있음을 알아 두자.

표 2를 보면 15가지 컬러가 지원되고 있음을 볼 수 있는데 여기서 3270-PC를 가지고 예를 든다면 High-Intensity 컬러(8~15)는 지원이 되지 않는다. 만일 여기에서 이러한 컬러들을 지정한다면 High-Intensity 컬러 대신 Low-Intensity 컬러들이 나타날 것이다.

• / BColor : 배경색상은 이 옵션에 의해서 지정이 된다. 범위는 0~7까지이며 정하지 않을 경우에는 처음에 말을 했듯이 DX가 시작될때의 커서위치의 컬러로 정해져 버린다.

```
dx /bcolor:1
```

• / FColor : 현재 화면상에서 볼 수 있는 컬러, 즉 전경색(Foreground Color) 을 이 옵션을 사용하여 조정할 수 있다. 범위는 0~15까지이며, 만일 지정되지 않으면 처음 DX가 시작되었을 때의 커서위치의 컬러로

〔표 2〕 DX에서 지원하는 컬러

스위치	컬 러
0	검정색(Black)
1	파란색(Blue)
2	초록색(Green)
3	청색(Cyan)
4	적색(Red)
5	자주색(Magenta)
6	갈색 또는 노랑(Brown or Yellow)
7	옅은 회색(Light Gray)
8	짙은 회색(Dark Gray)
9	옅은 파란색(Light Blue)
10	옅은 초록색(Light Green)
11	옅은 청색(Light Cyan)
12	밝은 적색(Bright Red)
13	밝은 자주색(Bright Magenta)
14	밝은 노란색(Bright Yellow)
15	밝은 백색(Bright White)

지정된다.

```
dx /fc:14
```

• / DColor : DX로 디렉토리를 보았을 때 High Intensity로 되어 있었음을 기억할 것이다. 바로 이 옵션을 이용하면 디렉토리의 컬러를 변경할 수 있다. 범위는 0~15이며, 만일 지정하지 않았을 경우는 ForeGround 컬러와 High Intensity 속성을 토글시켜 버린 결과를 가지고 디폴트로 컬러를 지정하게 된다. 예를 들면 전경색이 Yellow이면 디폴트 디렉토리 컬러는 Bright Yellow가 된다.

```
dx /dc:15
```

• / Output : 이 옵션은 출력을 화일로 만들거나 (>…I/O 재지정) 다른 프로그램의 입력으로 (|...파이핑)할 때 사용된다. 만약 어떤 한정자로써 :nul로 지정되면 모든 출력은 안보이게 된다.

```
dx /out >a:direct.dx
```

위의 예는 출력을 a : direct.dx라는 이름의 화일로써 저장하라는 의미이다.

• / PRint : 이 옵션은 출력을 프린터로 보내는 것으로써 DOS의 copy filename.ext prn 또는 copy filename.ext lpt1와 비교를 할 수 있다. 한정자로는 프린터의 기종을 써주는 것이 있는데, 만일 지정하지 않으면 EPSON과 IBM용으로 된다. 그리고 밝게 나타나는 디렉토리는 디렉토리는 두번 찍게 된다.

```
dx a:/pr
dx /print:ibm/out:b:myfile
```

(2) 화일매칭 옵션

• / Command : 이 옵션을 실행될 수 있는 화일들인 .EXE, .COM, .BAT 들만을 찾아 준다.

```
dx/c c:
```

●확장된 패턴매칭 기능

DX가 가지고 있는 기능중에서 어쩌면 가장 특이하고 막강한 능력이 바로 이 화일매칭 기능인데 이 구조는 여러개의 선택할 패턴과 또는 제외할 패턴 모두를 지정할 수 있다. 예를 들면

```
dx .dbf abc*.bas+ralph.
```

라는 것은 확장자가 .dbf로 끝나고, abc로 시작되는 베이직 화일, ralph라는 화일을 찾으라는 것이 된다. 여기서 만일 화일이름이나 확장자를 제외했을 경우는, '*'라고 가정을 하고 작업을 하게 된다. 즉 다시 말하자면 'FILES'과 'FILES.*'는 같은 표현이 되는 것이다. 마찬가지로 '.COM'과 '*.COM'은 같은 의미가 되며 세미콜론(;)은 어떤 패턴을 찾을 경우 제외할 패턴을 표시한다.

```
dx .bas+.asm;goodfile
```

위의 경우는 *.bas, *.asm을 찾는데 그중에서 goodfile.bas와 goodfile.asm은 제외를 하라는 말이 된다. 또 예를 들자면

```
dx ;test*;.com
```

위의 의미는 test로 시작되는 모든

화일 또는 확장자가 com인 모든 화일을 제외하란 의미가 된다. 이들 기능 외에도 DX는 각 패턴마다 다수의 와일드 카드문자 '*'를 쓸 수 있다. DOS에서는 '?'는 여러개를 쓸 수 있고 '*'는 1개 밖에는 쓰지 못하지만 DX에서는 양쪽 모두를 자유롭게 쓸 수 있다. 'A*Z'는 A로 시작하고 Z로 끝나는 패턴을 의미한다. '.*S'는 S로 끝나는 모든 패턴을 의미하는 것으로 *.pas, *.bas, *.s라는 모든 것이 선택되어진다. 또 한예로 '*TEMP*+.TMP'는 이름 어느 곳에든지 TEMP가 되어 있는 것과 확장자가 TMP로 끝나는 모든 것을 선택하게 되고 ';T*F?;.COM'는 T와 F가 들어 있는 패턴과 COM이라는 확장자를 가진 화일을 가진 화일을 제외한 모든 것이 선택된다는 의미이다. 'DX.COM+DX-HELP.COM' 은 2개의 화일을 선택한다는 의미이다.

●화일속성 매칭

DX는 화일의 속성이 무엇이든지간에 매칭에 포함시키거나 제외시킬 수 있다. 만약 정의가 되어지지 않으면 히든화일과 시스템화일은 제외시킨다. 화일의 속성을 제어하는 스위치는 다른 일반 옵션들과는 조금 다른데 이 스위치에는 어떠한 값이 아닌 3가지의 상태를 갖는다. 3가지의 상태에는 'only', 'include', 'exclude'가 있다. 이 상태에서 어떤 속성을 더할 경우 스위치 뒤에 '+'를 써주면 그 속성을 추가 시킬 수 있고, 만일 '-'을 써주게 된다면 그 속성을 제거할 수도 있다. 만일 only로써 스위치가 지정이 된다면 다른 속성은 체크되지 않는다. 그러나 /Full 스위치를 쓸 경우는 다음에 오는 어떤 스위치라도 무시되어 버린다.

- /Full : 이 옵션을 사용함으로써 가장 간단하게 모든 화일들을 볼 수 있다. 어떤 특수한 속성의 지정없이 모든 속성들을 포함한다.
- /Archive : 이 옵션을 지정하므로서 화일의 속성이 Archive인 화일들을 선택할 수 있다. DOS에서 화일을 만들거나 수정할 경우 DOS는 그 화일의 속성을 Archive로 만들어 놓는다.

```
dx/ff/a
```

백업되어 있는 현 디렉토리 상의 모든 화일을 보기 원할 경우에는 다음과 같이 쓸 수 있다.

```
dx/full/a+
```

옵션지정이 디폴트일 경우에는 Archive 속성이 자동으로 지정되지만 만일 DX를 Customize했을 경우에는 반드시 '/A+'라는 옵션을 주어야만 Archive 속성이 지정된다.

- /Directory : 이 옵션은 화면 출력에 있어서 디렉토리를 출력할 것인가 아닌가를 결정해주는 옵션이다.

dx / d : 디렉토리만을 출력하여 준다.

dx / d+ : Default시에 디렉토리를 포함하여 출력.

dx / d- : 디렉토리를 제외한 나머지를 출력.

dx / f / d- : 디렉토리를 제외한 모든 화일의 출력

- /Hidden : 이 옵션을 지정하므로서 히든화일들의 출력을 제어한다. 히든화일이란 숨겨진 화일이라고도 하며 IBMDOS.COM과 IBMBIO.COM, 그리고 유저들이 속성을 'Hidden'으로 만들어 놓은 것들을 의미한다.

dx / h : 히든화일들만 출력한다.

dx / h+ : 히든화일들을 포함하여 출력한다.

dx / h+ / s+ : 히든화일들과 시스템화일들을 포함하여 출력한다.

dx / h- : 디폴트 상태에서 히든화일들을 제외하고 출력

dx / f / h- : 히든화일들을 제외한 모든 화일이 선택되어 출력된다.

dx / f / h- / s- : 히든화일들과 시스템화일을 제외한 모든 화일을 선택하여 출력

- /Normal : DOS에는 없는 화일의 속성으로써, 아무런 속성을 가지고 있지 않는 DX만이 가지고 있는 속성이다.

dx / n : 'normal'이라는 속성을 가진 화일만을 출력한다.

- dx / n+ : Default시에 'normal'이라는 속성을 가진 화일들을 포함해서 출력한다.

dx / n- : 'normal'로 속성이 주어져 있는 화일들을 제외하고 출력한다.

dx / f / n- : 'normal'로 속성이 주어져 있는 화일을 제외한 모든 화일을 출력한다.

- /Readonly : 이 옵션을 지정하므로서 'readonly'로 특성이 주어져 있는 화일들을 출력할 수 있게 해주는 옵션이다. 'readonly'의 속성이 주어지면 그러한 속성이 주어져 있는 화일에 대해서는 어느정도 간단하게나마 프로텍트를 걸어서 보호할 수가 있다.

dx / r : 'readonly'라고 특성이 주어진 화일만을 출력한다.

dx / r+ : 디폴트시에 'readonly'화일을 포함해서 출력한다.

dx / r- : 'readonly'화일을 제외한 나머지 화일들을 출력한다.

dx / f / r- : 'readonly'화일을 제외한 모든 화일을 출력한다.

- /System : 시스템 화일들을 출력한다. 즉 IBMDOS.COM, IBMBIO.COM 등을 말하는데 이 화일들을 출력한다.

dx / s : 시스템 화일만을 출력

dx / s+ : 시스템 화일을 포함하여 출력한다.

dx / h+ / s+ : 히든화일, 시스템화일들을 포함하여 출력한다.

dx / s- : 시스템화일만을 제외하고 출력한다.

dx / f / s- : 시스템화일을 제외한 모든 화일을 출력한다.

dx / f / h- / s- : 히든화일, 시스템화일만을 제외한 모든 화일을 출력한다.

- /Size : 특별한 옵션으로 이런 기능을 가진 유틸리티는 거의 드물다고 볼 수 있는데 바로 원하는 화일의 크

기에 속하는 화일들만을 선택적으로 출력하여 볼 수 있는 기능이다. 여기에 들어가는 옵션으로써 어떤 범위를 나타내는 2개의 수가 들어가게 된다. 옵션에 '-'가 들어가면 그 값을 제외한 범위가 선택되어 진다.

dx / si : 1000 : 10000 : 화일들중 1000바이트에서 10000바이트 사이의 크기를 가진 화일들을 출력한다.

dx / si : -500 : 25000 : 화일들 중 500바이트보다 작거나 25000바이트보다 큰 화일들을 출력한다.

dx / si : 2345 : 2345바이트인 화일만을 출력한다.

dx / si : -45 : 45바이트가 아닌 모든 화일들을 출력한다.

• / DAte : 이 옵션 역시 다른 유틸리티에서는 볼 수 없는 독특한 기능이다. 다른 유틸리티에서 보아 왔던 것은 주로 화일의 분류에서나 날짜순으로 출력이 되는데 DX는 최종 업데이트 되어진 날짜, 즉 화일의 저장된 날짜를 가지고도 분류하여 출력할 수 있다. 물론 위의 크기에 의한 출력과 마찬가지로 어떤 저장된 시간범위내의 화일들을 선택하여 입출력할 수 있을 뿐만 아니라, 어떤 시간이후, 어떤 시간이전 등의 여러가지 선택 옵션들이 주어진다. 입력할 숫자(날짜)는 6자리의 수이다. 입력할 숫자의 양식으로는 'yymmdd'로 표시가 되는데, 여기서 각각의 의미는 다음과 같다.

yy : 년도위 뒷자리 숫자
mm : 월(1자리의 경우는 앞에 0을 붙인다)
dd : 일(역시 1자리의 수는 앞에 0을 붙인다.
를 나타낸다.

dx / da : 851020 : 851101 : 화일들 중 1985년 10월 20일에서 11월 1일 사이에 저장된 화일들을 선택하여 출력한다.

dx / da : -840310 : 850101 : 화일들 중 84년 3월 10일 이전 또는 85년 1월 1일 이후에 저장된 화일들을 선택하여 출력한다.

dx / da : 851115 : 오직 1985년 11월 15일에 저장되어진 화일만을 출력한다.

dx / da : -840523 : 1984년 5월 23일 이전에 저장되어진 화일들을 출력한다.

그러나 / DAte 스위치는 여기에 몇가지 더 추가가 되는 것이 있다. 이것 역시 이 기능을 더욱더 독특한 특성으로 꾸며주는 옵션들인데 t, y, m, w 들이 그것이다.

t : Today
y : Yesterday
m : Month(30 : 0) 즉 30일로 정의
w : Week(6 : 0) 6일로 정의 되어짐

이것으로써 오늘, 어제, 지난주, 지난달 만들어진 화일들만을 선택적으로 출력할 수 있다.

dx / da : t : 오늘 저장되어진 화일만을 출력한다.

dx / da : -y : 어제 저장되어진 화일만을 출력한다.

dx / da : 5 : y : 화일들 중 5일전에 저장되어진 화일과 어제 저장되어진 화일만을 선택하여 출력한다.

dx / da : m : 화일들 중 지난달에 저장되어진 것을 선택하여 출력한다.

dx / da : -w : 화일들 중 지난주 저장되어진 화일들을 출력한다.

• / Time : 이 옵션 역시 위의 날짜에 관한 예처럼 아주 재미있는 옵션이다. 위에서 2개 년도의 입력처럼 시간옵션이 주어져야 한다. 시간을 주는 옵션은 1자리의 숫자에서 6자리의 숫자가 입력이 되는데 시간에 주어지는 형식으로는 'hhmmss' 또는 'hhmm'으로 주어지게 된다. 여기서 hh는 24시간형식인데 만일 12시간으로 옵션을 주려면 앞에서 기술한 '/12'이란 옵션을 주어야 한다. 여기서 mm은 분, ss는 초를 가리킨다. 그리고 숫자의 입력시 4자리 숫자보다 작게되면 나머지 앞자리는 0이 더해지게 된다. 그리고 이 옵션은 '/-DAte' 옵션과 조합하여 사용할 수 있다.

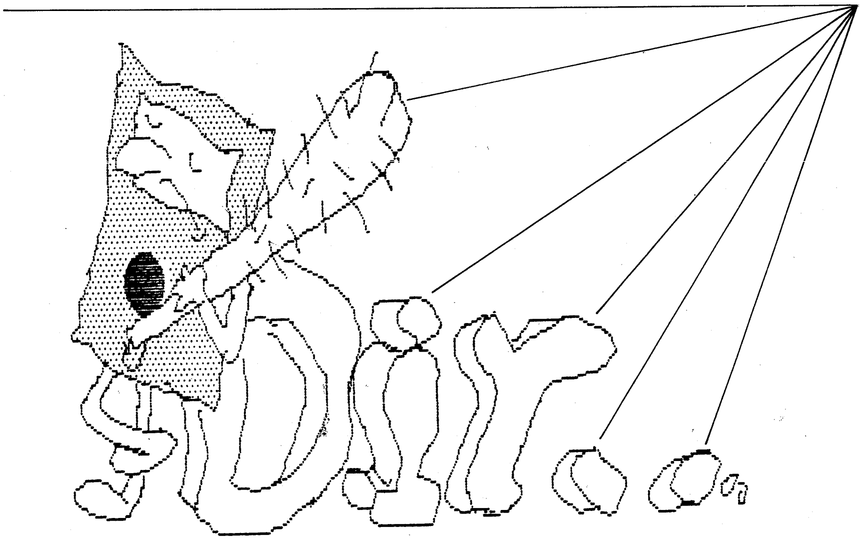
dx / ti : 100 : 0530 : 01시부터 05시 30분 사이에 저장되어진 화일을 출력한다.

dx / ti : -0900 : 2300 : 09시 이전 혹은 23시 이후에 저장되어진 화일을 출력한다.

dx / ti : 1234 : 12시 34분에 저장되어진 화일을 출력한다.

dx / ti : -1000 : 10시에 저장되어지지 않은 화일을 출력한다.

• / Dot : DOS에서 어떤 디렉토리 안에 있다면 현재 그 디렉토리를 표시하는 '.'과 바로 위의 상위 디렉토리를 표시하는 '..'를 볼 수 있는데 이 옵션



이 주어짐으로 인해서 이것들을 볼 수도 안보이게도 할 수 있는 옵션이다 이 옵션은 앞에서 기술한 / Directory 옵션과 연관지어서 생각할 수 있는데 만일 디렉토리를 안보이게 'DX / d－'로 옵션이 주어질 경우 이 / Dot 옵션은 아무 효력을 발생시키지 못하게 된다.

(3) 유틸리티 옵션

• / SOrt : DX가 가지고 있는 옵션 중에는 고급 유틸리티에서나 볼 수 있는 기능이 다수 제공되는데 이것도 그중의 하나이다. 일단 / SO 옵션을 뒤에 그뒤에 어떤 원하는 작용을 하게 하는 한정자를 줌으로써 더욱 더 세련된 기능을 맛볼 수 있다. 한정자로는 5개가 있는데

N : 이름으로 분류
E : 확장자로 분류
S : 화일 크기로 분류
D : 날짜로 분류
S : 시간으로 분류

그리고 이것을 더욱 멋있게 꾸며주는 것으로써 '＋', '－'이 있는데 '＋'옵션을 쓰면 정순(Ascending)으로 분류를 해주고 '－'옵션을 쓰면 역순(Descending)으로 분류를 해준다.

dx / so : 한정자가 없을 경우, 화일의 이름과 확장자를 정순으로 분류한다.

dx / so : ne : 화일의 이름과 확장자를 정순으로 분류한다.

dx / so : e－n－ : 확장자에 의해 역순으로 분류하고, 또 화일 이름에 따라 역순으로 분류한다.

dx / so : snd－ : 화일 크기에 따라 정순으로 분류하고 화일 이름에 의해 정순으로 분류한 후 저장시간에 따라 역순으로 분류한다.

• / FFind : 이 옵션으로 Advanced Norton Utility안에 있는 화일을 찾는 유틸리티인 ff.exe와 비슷한 기능을 DX에서 맛볼 수 있다. 그러나 어떤 시간적인 면을 제외하고 DX의 여러 가지를 조합한 기능과 비교를 한다면 ff.exe는 비교가 될 수 없을 것이 만일 경로(path)가 지정되지 않으면 DX는 현재 디렉토리에서 주어진 임무를 수행한다. 그러나 다른 디렉토리에서의 수행을 원한다면 경로를 지정해 주어야 한다.

dx\ / ff : 모든 디렉토리에서 화일을 찾는다.

dx .pas / ff : 확장자가 .pas로 끝나는 화일을 찾아준다.

dx d : \test.c / ff : d: 드라이브의 test.c라는 화일을 찾는다.

• / FSize : 이 화일크기 옵션은 어떤 화일을 다른 디스크로 전송할 때 전송을 할 수 있는가 없는가의 유무를 판단한다. 이 옵션을 사용하여 어떤 결과를 얻었다 하더라도 실제는 아무런 전송이 이루어지지 않는다. 만일 충분한가 아닌가를 단순한 메시지로 볼 경우는 / FSBlink 옵션을 사용한다. / FSize 옵션에서 어떤 값을 변수로써 사용하는데 이것은 올바른 디스크 드라이브 이름과 디스크드라이브의 최소한의 할당량(클러스터)의 크기이다.

dx ＊.pas / fs : a : ＊.pas라는 확장자를 가진 화일이 A 드라이브로 전송될 수 있는지를 알아본다.

dx b : \＊test / fs : c / ff : test로 끝나는 모든 화일이 C 드라이브로 전송될 수 있는가를 알아본다.

dx xxx. / fs : 512 / fsb : xxx라는 화일이 512 바이트의 최소한의 할당으로 전송하는데 있어서 얼마마한 공간이 필요한가를 알아낸다. 그리고 / fsb 옵션을 사용하였으므로 메시지가 깜박거리면서 나타난다.

• FSBlink : 이 옵션을 사용하면 그 결과로써 충분한 공간이 있을 경우 모니터의 맨 아랫줄에 'SUFFICIENT'라는 문자가 깜박거리게 된다. 만일 충분한 공간이 없을 경우는 'NOT SUFFICIENT' 라는 문자가 깜박거리게 될 것이다. 이 옵션은 위의 / FSize 옵션과 같이 사용하여 효과를 얻을 수 있다.

```
dx /fs:c/fs
```

• / Ldir : 이 옵션을 사용함으로써 디스크에서 찾을 수 있는 디렉토리의 이름을 출력한다. 만일 경로(path)가 지정되지 않으면 현재의 디렉토리를 기준으로 하여 실행된다.

dx c : / ld : 드라이브의 모든 디렉토리를 출력한다.

dx \ / ld / w : 루트 디렉토리에서 시작하여 모든 디렉토리를 Wide 형식으로 출력한다.

• / CKsum : 이 옵션은 주로 원래의 화일과 복사되어진 화일의 체크섬을 비교할 때 이용된다. 만일 체크섬이 일치된다면 두 화일은 같다고 판단할 수 있다. 이 방법은 화일이 변경되었을 경우, 재빠른 확인을 위해 사용된다.

```
dx \/ffind/cksum/output:prn
```

위의 예는 체크섬과 함께 모든 화일의 완전한 목록을 프린터로 찍으라는 예이다. 먼저 DX는 루트 디렉토리에서 시작하여 모든 화일에 대하여 체크섬을 하고(/ CKsum), 그리고 프린터를 통하여 출력한다(/ Output). 체크섬은 4자리의 16진수로 출력되는데 그 숫자는 원래의 것과 일치하는한 중요하게 여기지 않아도 된다.

• / DElete : DX에서 이 옵션을 줌으로 해서 어떤 원하는 패턴과 일치되는



모든 화일을 지워 버린다. 만일 어떤 패턴이 정해지지 않으면(Default) DX는 각각의 화일을 지우기 전에 확인하게 된다. 여기서 사용자가 주의해야할 일은 가령 하드디스크를 사용할때 /FFind 옵션과 같이 혼용하여 사용할시 모든 하드디스크 내의 화일을 지울 수 있다는 점이다.

```
dx b:/del
```

위의 예의 경우는 B드라이브의 모든 화일을 지우라는 의미이다. 이때 DX는 각각의 화일을 지울때 확인을 하게된다.

- /FAttr : 이 옵션을 사용함으로써 어떤 패턴과 일치되는 모든 화일의 속성을 바꿀 수가 있다. 만일 패턴이 주어지지 않는다면 DX는 각각의 화일에 대하여 속성을 바꿀때 확인을 해 온다. 이 스위치의 내용은 화일에게 주어질 속성을 말하며 그 종류는 다음과 같다.

A : Archive
H : Hidden
N : Normal
R : ReadOnly
S : System

이 스위치는 '+'를 지정함으로써 작동을 하게 되고, '−'를 지정함으로써 작동이 중단된다. 앞에서 언급을 했듯이 Normal이라는 화일 형태는 아무 속성이 없는 특수한 형태를 말한다. 어떠한 화일에 속성을 주기 위해서는 앞의 스위치 내용에 '+'를 붙여서 기술을 해주면 된다. 숨겨진 화일과 시스템 화일로 만들기 위해서는 "/H+/S+"로 써주면 된다.

dx \dira*.* / fa : a+ : dira라는 디렉토리의 모든 화일을 archive 속성으로 갖도록 한다.

dx b : / fattr : nr : B드라이브의 모든 화일을 readonly로 만들고 나머지 기존의 속성들을 클리어시킨다.

dx \ / a / fa : a− : 속성이 Archive로 되어 있는 화일의 속성을 나타내는 스위치를 OFF시킨다.

dx ibm*.com / full / fa : n : IBM으로 시작되어지는 화일 즉 시스템 화일의 속성을 'normal'로 바꾸어준다.

만약 여기서 지울 수 없는 속성을 가진 화일들을 /DELete 옵션을 사용하여 지울 경우는 먼저 속성을 normal로 만들어 준 다음 지우면 된다.

```
dx a:readonly.fil/fa:n/del
```

위와 같이 하면 readonly화일을 지울 수 있다.

- /Verify : 위의 /DELete 옵션과 /Fattr 옵션에서 어떠한 조건없이 위의 옵션을 실행할 경우 각각의 화일에 대하여 수행을 하기 전에 확인을 하게 되는데 이것이 바로 이 /Verify 옵션 때문이다. 일반적으로 이 옵션은 수행이 되고 있다. 만일 이 옵션이 필요가 없다면 '/V−'라는 스위치를 줌으로 해서 확인절차를 생략해 버린 채 작업을 수행할 수 있다. 예를 들면 시스템 디스크를 데이터 디스크로 바꾸어 주려면 시스템 디스크안에 있는 시스템 화일들을 지워버려야 하는데 이럴 경우 다음과 같은 절차를 거치면 된다.

```
dx b:ibm*.com+command.com/f/fa:n/del/v-
```

조금 형식이 길어서 어렵게 느껴지지만 익숙하게 되면 효율적으로 작업을 수행할 수 있다. 일반적으로 BAK, TMP라는 확장자의 화일은 모두 확인절차를 생략한 채 지울 수도 있는 것이다.

〔표 2〕 DX의 배치 수행에 쓰이는 템플리트 변수(1)

템플리트 변수	의미(예)
SD	드라이브 지정("C:")
SF	화일이름("Myfile")
SE	확장자(".Ext")
SN	화일이름과 확장자("Myfile.Ext")
SP	화일의 경로(" utility/Misc")
SS	완전한 화일에 대한 서술("C: utility/Misc/Myfile.Ext")

(4) 배치화일 운영

DX의 특이한 기능 중에는 또 배치화일 오퍼레이션 기능이 있다. 이 기능은 DX의 화일매칭 능력을 이용하는 것으로 이렇게 해서 만들어진 화일은 다른 디스크에 존재할 수도 있으며 키보드로부터 입력을 받을 수도 있고 다른 프로그램으로부터 파이프 처리될 수도 있다.

표 3는 임의의 디렉토리안의 화일에 대한 적용예와 함께 각 템플리트 변수(Template Variable)를 보여주고 있는데 이들 변수들은 각각이 대입된 내용으로 대용되어 사용될 수 있다.

쉬운 예로 dx / Batch 지정을 이용하여 다음의 명령을 실행시켜 보기 바란다. 또는 이 명령을 내용으로 하는 텍스트화일을 만들어 놓고 'dx / Batch : 〈화일이름〉'으로 실행시킬 수도 있다.

```
copy $n b:$f.hld
```

배치화일을 생성시킬 때 배치 템플리트(Template)안의 모든 행은 각각 그 행이 수행됨에 따라서 결과를 가져온다. 즉 각 화일이 일치됨에 대하여 하나 이상의 프로그램을 수행할 수 있다는 것을 말한다. 또 배치화일의 처음과 끝에는 '${'과 '$}'이 있어야 하는데 한 예를 본다면 매칭이 되는 모든 화일을 B드라이브로 복사를 하고 비교를 할때 DOS의 파이프처리(|)를 사용하고자 하는 경우는 시작과 끝부분 그리고 에코(eche)를 OFF시킬 때의 정지부분에 이 표기를 위치시켜 사용할 수 있다.

```
${echo off
${echo Insert archive disk in
        B: and strike any key ...
${pause
copy $n b:
echo n ¦ comp $n b:
$}echo Copy and Compare complete...
$}pause
```

모든 템플리트 변수는 한행에서 처음 두글자만을 인식하는 '${…$}'를 제외하고 행의 어느곳에서나 인식한다. 이 배치화일을 실행시키기 위해서는 앞에서도 잠깐 언급했듯이 에디터를 사용하여 아스키코드 형태로 저장을 한다음 /Batch 스위치를 사용한다.

- /Batch : 이 스위치는 배치 수행을 하도록 하며 DX로 하여금 템플리트 화일을 읽도록 해준다. 만약 특별한 화일에 대한 기술이 없으면 DOS의 표준 입력인 키보드 입력, 또는 입력의 재지정(Redirection), 파이프처리의 입력 등을 읽게 된다. 간단하게 DOS의 echo명령에 대한 한 라인의 배치실행을 예로 들면 echo 명령은 DX에 대해 파이프 처리가 가능하다.

```
echo ren $n $f.hld ¦ dx *.xx.tmp/b
```

위의 예는 화일명이 xx로 끝나고 확장자가 TMP로 끝나는 모든 화일을 확장자가 HLD인 화일로 이름을 바꾼다. 여기서 사용되어진 '¦'를 파이프(pipe) 라고 한다.

어떤 특별한 문자는 DOS에 의해서 해석되어 지는 이유로 echo에서 DX로 보내는데 이용할 수 없는 경우가 있는데 이러한 제한조건을 극복할 수 있도록 표 4에서와 같이 또 다른 형태의 템플리트 변수들을 준비하고 있다. 아래의 예는 PROG1, PROG2를 실행하는 배치화일을 수행하면 각 화일마다 *.IN으로 일치가 되는 화일을 프린터로 출력하게 된다.

```
echo prog1 $n $_prog2 $n $_print
        $f.prn ¦ dx *.in/b
```

〔표 4〕 DOS가 인식하는 특별한 문자를 대용할 수 있는 템플리트 변수(2)

템플리트 변수	의 미
$	Carriage Return/Line feed echo를 통해 다수의 행처리가 가능토록 해준다.
$(	'<' 배치화일 내에서의 입력 방향 재지정을 허가한다.
$)	'>' 이미 존재하는 화일에 대한 배치화일의 겹쳐 쓰기에서 출력의 방향을 허용한다.
$!	'¦' 배치화일내에서의 파이프 처리를 허가한다.
$$	'$' $기호를 쓸때는 구분을 위해 2개의 연속된 $$가 필요하다.

그러나 DOS의 COMP 또는 EDLIN 같은 표준 입력을 읽는 배치 화일내의 코맨드를 쓸경우 입력이 재지정되거나 파이프인이 되지 않는 한 위의 방법을 사용할 수 없다. 만일 프로그램에서 이러한 DOS의 표준 입력을 읽으려고 했을 경우는 그것에 대하여 아무 것도 남지않을 뿐만 아니라 다시 재부팅을 시켜야 한다. 그러나 다음과 같은 방법을 사용할 경우에는 위의 문제점을 극복할 수 있다.

*디스크상의 배치 템플리트를 사용했을 경우

* /Batch 스위치에 화일이름이 주어진 경우

이때의 확장자가 지정되지 않으면 '.TPL'로 된다.

```
dx myfile.*/batch:sample.tpl
```

*배치화일을 수행시킨 후 즉시 실행중지를 시킬 경우(/EXec 스위치 사용)

*해당 프로그램 내에서 입력을 지원해줄때(키보드 입력으로부터 입력을 마칠 때는 Ctrl-Z을 누른다.)

- /EXec : 이 옵션은 실행이 즉시되느냐 아니냐를 결정해준다. 이 스위치를 지정하지 않을 경우는 ON되어 있는 상태가 된다. 이 스위치를 OFF시킨 경우를 보면

```
dx pattern/b:template/ex-
```

- /CD : 배치화일 내에서 디렉토리의 변경을 해줄 수 있도록 하는 스위치이다.

```
dx files/batch:skeleton.txt/cd
```

(5) DX의 키입력을 쉽게

DX는 많고도 강력한 기능들이 많은 반면에 그것을 쓰기 위해서는 매번 그 스위치를 입력해 주어야 하는 불편이 있는데 어느 정도는 편리하게 단순화 시킬 수 있다. 스위치를 정의하지 않은 상태에서 2가지의 방법을 DX는 제공한다. 즉 DX는 DOS환경내에서 2가지의 스트링을 인식할 수 있다. 바로 DX와 DX0의 두가지이다. DX0 같은 경우는 컴퓨터의 처음 전원을 ON시킴과 동시에 실행되는 AUTOEXEC.BAT 화일 내에 미리 옵션들을 주는 방법이다. 다른 또 하나의 방법은 기존의 사용그대로 모든 옵션들을 지정해 주는 방법이다. 다음을 지정하고 배치화일에서 실행되는 예를 살펴보기 바란다.

실행에서는 다음의 예처럼 세트를 시켜줌으로 해서 실행될 수 있다.

```
set dx0=/switch/switch....
```

또는

```
set dx=/switch/switch.....
```

위의 예는 머리말을 한줄 출력시키고 화면이 가득차면 키의 입력을 기다리며 화면의 문자 출력을 소문자로 하고 정해지지 않는 화일의 이름일 경우 D : $$DX.BAT로 정해 준다음 출력을 정지한다. 배치 화일에서의 실행이 아닌 코맨트 라인의 처리에 있어서는

'//'로 시작하게 되면 배치화일내에 정의되어 있던 옵션들은 모두 무시된다.

```
dx a://
dx //w test.fil
```

(6) 참고사항

●실행종료 조건

DX는 종료를 하게 되는 조건을 비롯하여 여러 조건들을 나타내는 'Error-Level' 변수들을 세트시킨다. 이것은 배치화일 내에서 에러를 발견하는데 커다란 도움이 된다. 다음의 조건에 의해서 변수의 값이 돌아오게 된다.

0 : 정상 종료
10 : 일치가 되는 화일이 없는 경우
20 : 전송을 하는데 필요한 충분한 공간이 없는 경우
30 :^C에의한 강제 멈춤의 경우
40 : 문법 에러가 발견되었을 경우
50 : 디스크 액세스 에러. 잘못된 드라이브의 지정 또는 오픈할 화일이 없는 경우
100 : 런타임에러 또는 관리 가능한 메모리의 범위를 벗어 났을 경우

예를 들어보면

```
dx /h/out:nul
if errorlevel 10 goto no-hidden
```

●실행속도에 대하여

DX를 램디스크에서 실행한다면 DOS의 DIR 명령보다 7%가 빨라질 수 있다. 하드디스크에서의 로드시에는 0.66초에서 1초가 걸리며 플로피디스크를 사용한다면 약 3.5초가 걸린다. 만일 8087, 80287 등의 보조 프로세서를 사용한다면 2.3%에서 6%가 빨라진다. 소문자의 경우 2% 정도, 정의가 되지 않을 경우, 즉 첫자만을 대문자로 한 경우는 5% 정도의 시간이 더 걸린다.

• /INfo : 이 옵션을 지정하면 DX에 대한 도움말을 얻을 수 있다.

(7) 요약

다음은 이제까지 살펴본 DX의 파라미터 및 각 스위치를 요약하여 나타낸 것이므로 숙달될 때까지 참고로 하기 바란다.

•포매팅 옵션

/Pause : 화면이 차있을 때 출력을 정지하여 키입력을 기다린다.
/Wide : dir / w와 같이 한줄에 5개의 화일을 출력한다.
/SINgle : dir과 같이 한줄에 1개의 화일을 출력한다.
/Totals : 전체의 남은 용량 등 모든 정보를 출력한다.
/Header : n : n(0..2) 줄의 머리말을 출력
/Stats : n : 디스크에 대한 통계를 출력. n(0..4) 가지의 정도로 분류해서 출력.
/Case : 화면에 출력되는 절차의 종류를 선택함.
/12 : 시간 출력을 12시간 형식으로 출력.
/Out put : 재지정과 파이프 처리에 의한 출력.
Output : filespec : 화일로 출력
/PRint / printer : 프린터로 출력
/FSBlink : /FSize 옵션에서 메시지를 점멸시키기 위해 사용.

●화일매칭 옵션

/Command : 실행가능한 화일을 찾아준다.(.com, .exe, .bat)
/SIze : low : high : 크기가 low에서 high 사이의 크기를 가진 화일을 찾아 출력.
/SIze : -low : high : low보다 작거나 high 보다 큰 화일을 찾아 출력
/DAte : -yymmdd : yymmdd : 어느 기간전, 기간이후 저장된 화일을 찾아 출력
/TIme : hhmm : hhmm : 어느 일정한 범위의 시간에 저장된 화일을 찾아 출력
/DOt : 디렉토리 안의 '.', '..'의 출력을 제어
/Full : 모든 화일을 출력
/Archive : archive 속성을 가진 화일을 출력
/Directory : 디렉토리를 출력한다.
/Hidden : 히든화일을 출력한다.
/Normal : 속성이 normal인 화일들을 출력한다.
/System : 시스템 화일을 출력한다.(ibmbio. com, ibmdos.com)
/Full / xxxx- : 지정되어 있는 속성을 제외한 모든 화일을 출력

●유틸리티 옵션

/Sort : nesdt- : 'nesdt-'에 의하여 화일을 분류한다.
/FFind : 다수의 디렉토리에서 화일을 찾는다.
/FSize : nnnn : nnnn이라는 클러스터의 크기를 가지고 다른 드라이브로 전송이 가능하지 체크한다.
/Ldir : 디렉토리 이름만을 출력
/CKsum : 화일을 체크섬 한다.
/DElete : 화일을 지운다.
/FAttr : ahnrs : 화일의 속성을 바꾼다.
/Verify- : 확인을 묻는 절차를 생략한다.

●배치화일 생성

/Batch : I/O 재지정과 파이프처리가 되는 DOS의 기본 입력으로부터 템플리트를 읽는다.

Prototype File Variables
$D : 드라이브의 이름 지정
$F6 : 화일이름
$E : 확장자
$N : 화일 이름과 확장자
$P : 경로지정
$S : 드라이브 이름, 경로, 화일이름, 확장자 지정
$_ : 복귀 개행(CR, LF)
$(: "<" 입력 방향지정
$) : ">" 출력 방향 지정(겹쳐 쓰기, overwrite) $! : "|" 파이프 처리
$$: 출력에서 $를 사용할 때
$} : 배치 화일의 말미로써 사용 그 줄의 맨앞 2글자
.Ctrl Break 또는 Ctrl-C : DX의 실행을 중지
.ESC : 전체적 통계를 건너 뜀

쿼트로의 폰트를 터보 파스칼에서 이용한다

심성호/명지대학교 전자공학과

터보파스칼 4.0에서 가장 획기적으로 변화한 것은 그래픽 부분일 것이다. 그래픽용 벡터 폰트가 4개나 들어있고 다양한 그래픽 지원 프로시듀어들을 사용할 수 있다는 점은 터보 파스칼의 기능을 더욱 돋보이게 한다. 여기에 쿼트로에서 사용하는 벡터 폰트마저 사용할 수 있다면 금상첨화가 아닐까. 아래의 방법을 사용하면 약간의 조작만으로 10개나 되는 벡터 폰트를 사용할 수 있다.

머리말

터보 파스칼 4.0이 나올 당시 모든 사람들은 그 매력에 반하지 않을 수 없었다. 편리해진 내장 에디터, 빨라진 컴파일 속도, 그리고 다양한 기능 등이 우리들을 즐겁게 하였다. 특히 기존의 것이(버전 3.0이전) 특정 그래픽카드(CGA, EGA 등)만을 지원하는 반면 4.0 이후로는 모든 그래픽카드를 지원하고 있어 어떤 컴퓨터이든지 호환성있는 프로그램을 작성할 수 있게 되었다. 터보 파스칼 4.0에서 획기적으로 변화한 것은 많지만 가장 획기적인 것은 그래픽 부분일 것이다.

특히 그래픽용 벡터 폰트가 들어 있다는 것과 다양한 그래픽 지원 프로시듀어들은 괄목할 만한 것이다. 심지어 어떤 이들은 터보 파스칼 4.0은 정통 파스칼에서 벗어난 '파스칼의 이단'이라고 평하기도 한다. 이렇듯 발전한 터보 파스칼에는 앞에서도 언급했듯이 그래픽용 벡터 폰트가 4개 들어 있다. 그런데 같은 회사인 볼랜드사에서 나온 쿼트로에서는 10개의 벡터 폰트를 사용할 수 있게 되어 있다. 그러면 터보 파스칼에서도 이 10개의 폰트를 사용할 수는 없을까.

여기에 소개하는 방법은 터보 파스칼안에 있는 방법인데 약간의 조작으로 10개나 되는 벡터 폰트를 사용할 수 있도록 하는 것이다.

터보 파스칼을 실행할 경우 그래픽 드라이버나 벡터 폰트들은 그때 그때 필요할 때마다 불러서 사용하게 된다. 그런데 유틸리티를 이용하여 프로그램과 같이 링크시켜 버리면 다음에 실행할 때 램안에 상주하여 다시 디스켓에서 호출하는 불편한 일이 없어지게 된다. 대신에 프로그램의 용량은 드라이버를 포함하는 만큼 커지게 된다.

필요한 화일들

필요한 화일들은 다음과 같다. 다음에 나오는 화일들을 디스켓 한 장에 모두 복사하여 둔다. 이 중에서 ＊.CHR은 쿼트로에서 복사한 것이고 MAKEOBJ.BAT는 리스트 1과 같다. FONTS.PAS는 터보 파스칼 4.0의 Graphics & Microcalc 디스크에 담겨 있는 화일인데 여기서 10개의 폰트를 사용하기 위하여 리스트 2와 같이 몇가지를 추가하였다.

BINOBJ.EXE: 바이너리 화일을 목적 화일로 바꾸어 주는 유틸리티(Disk 3 : Utility & Examples Disk)

＊.CHR: 쿼트로는 모두 4장으로 구성되어 있는데 그 중에서 10개의 ＊.CHR만 복사한다.

MAKEOBJ.BAT: ＊.CHR 화일을 ＊.OBJ로 만들어 주기 위한 일괄처리 화일로 리스트 1과 같으며 에디터를 이용하여 작성한다.

FONTS.PAS: FONTS.TPU를 만들어 주는 소스 프로그램. 이것은 리스트 2와 같이 기존의 FONTS.PAS에 새로운 폰트 화일을 삽입한다.(Disk 2 : Graphics & Microcalc Disk)

TPC.EXE: 터보 파스칼 4.0 명령어 라인방식의 컴파일러(Disk

〔리스트 1〕 MAKEOBJ.BAT

```
binobj trip.chr trip TriplexFontProc
binobj litt.chr litt SmallFontProc
binobj sans.chr sans SansSerifFontProc
binobj goth.chr goth GothicFontProc
binobj tscr.chr tscr TriplexScriptFontProc
binobj simp.chr simp SimplexFontProc
binobj scri.chr scri ScriptFontProc
binobj lcom.chr lcom ComplexFontProc
binobj euro.chr euro EuroStyleFontProc
binobj bold.chr bold BoldFontProc
del *.chr
```

1 : Compiler Disk)

TURBO.TPL : 프로그램 라이브러리. 여러개의 유니트(Unit)가 들어 있으며 이것은 TPUMOVER.EXE로 삭제 및 삽입이 가능하다. 경우에 따라서는 여기서 만드는 FONTS.TPU도 이 화일 안에 삽입 가능하다.

FONTS.PTU 화일을 만드는 방법

앞에서와 같이 하나의 디스켓에 필요한 화일들을 복사했으면 FONTS.TPU화일을 만든 작업에 들어간다.

1. 먼저 BINOBJ.EXE를 이용하여 *.CHR을 *.OBJ로 만들어 준다. 이것은 이미 일괄처리화일로 만들었으므로 MAKEOBJ. BAT를

```
A:\>MAKEOBJ
```

실행하면 된다. 이때 *.CHR 화일은 지워지고 *.OBJ화일이 만들어진다.

2. FONTS.PAS를 컴파일하여 TPU화일을 만든다. 위의 1번 과정이 정상적으로 끝나면 다음과 같이 명령한다.

```
A:\>TPC FONTS
```

이후로 정상적으로 끝났으면 에러 메시지없이 끝날 것이다. 에러가 생기면 에러 메시지가 가리키는 곳을 수정한다.

화면 1은 위의 과정이 정상적으로 실행되는 것을 보이고 있으며, 이 방법 외에도 유틸리티 디스크 안에 들어 있는 MAKE. EXE를 이용하는 방법이 있으나 여기서는 생략하기로 한다. 이제 FONTS.TPU라는 화일이 만들어졌다.

문자 표현 방식

명조체나 고딕체와 같이 글씨의 모양을 폰트(Font)라고 한다. 컴퓨터에서 문자를 나타내는데는 두가지의 방식이 사용된다. 하나는 비트맵 방식인데 데이터의 각 비트(Bit)가 문자의 각 화소에 해당한다. 다른 하나는 벡터 방식인데 좌표에 의해서 글씨의 모양이 결정된다. 참고로 **그림 1**을 보면 이해하기가 쉬울 것이다.

비트 맵 방식은 데이터가 작은 반면 특정 크기로 제약되어 있어 워드프로세서 등에서 사용된다. 벡터 방식은 데이터

〔리스트 2〕 쿼트로에 쓰이는 6개의 폰트를 추가시킨 예

```
{ Copyright (c)1985,87 by Borland International,Inc. }

unit Fonts;
{ Sample unit to accompany GRLINK.PAS. This unit links
  all the BGI graphics fonts into a single TPU file.
  This makes it easy to incorporate the font files
  directly into an .EXE file. See GRLINK.PAS for more
  information.
}
interface

procedure SansSerifFontProc;     ┐
procedure SmallFontProc;         │ 기존의 폰트
procedure TriplexFontProc;       │
procedure GothicFontProc;        ┘
procedure BoldFontProc;          ┐
procedure EuroStyleFontProc;     │
procedure ComplexFontProc;       │ 쿼트로에 들어 있는 폰트
procedure ScriptFontProc;        │
procedure SimplexFontProc;       │
procedure TriplexScriptFontProc; ┘

implementation

procedure SansSerifFontProc; external;
{$L SANS.OBJ }

procedure SmallFontProc; external;
{$L LITT.OBJ }

procedure TriplexFontProc; external;
{$L TRIP.OBJ }

procedure GothicFontProc; external;
{$L GOTH.OBJ }

procedure BoldFontProc; external;
{$L BOLD.OBJ }

procedure EuroStyleFontProc; external;
{$L EURO.OBJ }

procedure ComplexFontProc; external;
{$L LCOM.OBJ }

procedure ScriptFontProc; external;
{$L SCRI.OBJ }

procedure SimplexFontProc; external;
{$L SIMP.OBJ }

procedure TriplexScriptFontProc; external;
{$L TSCR.OBJ }

end.
```

가 비교적 길어진 대신 문자의 크기를 임의로 확대할 수 있다. 일반적으로 벡터 폰트는 닥터 할로우나 페인트브러쉬와 같은 그래픽 패키지에서 많이 사용된다. 한글 폰트를 보면 보통 사용되는 워드프로세서(보석글, 팔란티어, 바른글)는 비트 맵 방식을 사용하고 있고 매킨토시에서 사용되는 한글은 벡터 폰트를 사용한다.

FONTS.TPU화일의 이용법

파스칼을 좀 다루어 본 사람이면 리스트 3의 예제 프로그램을 쉽게 이해할 수 있을 것이다. FONTS.TPU 화일은 선언부의 uses문에 의해 정의된다. 단, 디렉토리 지정 메뉴에서 유니트 디렉토리는 GRAPH.TPU와 FONTS.TPU화일이 들어 있는 디렉토리를 가리켜야 한다.

다음은 RegisterBGIfont 프로시듀어를 액세스하는 것인데 리스트 3의 8행에서 35행까지를 말한다. 이것은 프로그램을 작성하는 선언부 다음에 항상 삽입하여야 한다.

다음은 일반적인 그래픽 함수를 사용하는 것과 동일하다.

그래픽함수 요약

그래픽함수는 매뉴얼을 보면 자세하게 나와 있지만 여기서는 벡터 폰트를 사용하는 것을 중심으로 설명하겠다.

● **SetTextJustify(Horiz, Vert: word)**

이 함수는 OutTextXT나 OutText의 함수를 사용할 경우 텍스트 출력방식을 정의한다. 지정 파라미터는 다음과 같다.

Horiz: 0…왼쪽 좌표
1…중앙 좌표
2…오른쪽 좌표
Vert: 0…하단좌표
1…중앙좌표
2…상단좌표

● **SetTextStyle(Font: word; Direction: word; CharSize: CharSizeType)**

현재 사용할 문자의 스타일을 선택

〔화면 1〕 MAKEOBJ 배치화일이 정상적으로 실행되는 과정

```
A>makeobj

A>binobj trip.chr trip TriplexFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

7213 bytes converted.

A>binobj litt.chr litt SmallFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

2126 bytes converted.

A>binobj sans.chr sans SansSerifFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

5452 bytes converted.

A>binobj goth.chr goth GothicFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

8489 bytes converted.

A>binobj tscr.chr tscr TriplexScriptFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

7295 bytes converted.

A>binobj simp.chr simp SimplexFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

4787 bytes converted.

A>binobj euro.chr euro EuroStyleFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

2557 bytes converted.

A>binobj bold.chr bold BoldFontProc
BIN to OBJ Converter Version 1.0  Copyright (c) 1987 Borland International

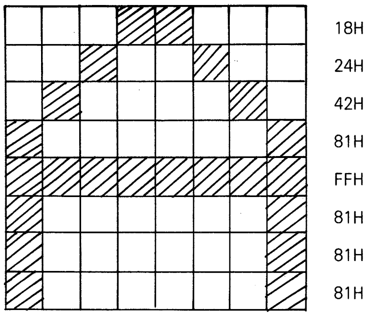
5125 bytes converted.

A>del *.chr

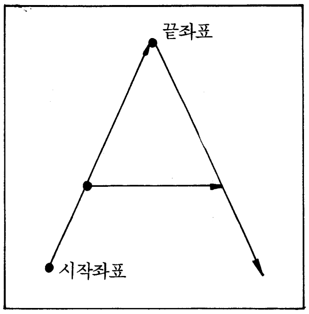
A>tpc fonts
Turbo Pascal  Version 4.0  Copyright (c) 1987 Borland International
FONTS.PAS(54)
54 lines, 22.5 seconds, 50720 bytes code, 0 bytes data.

A>
```

〔그림 1〕 문자표현방식



(a) 비트맵방식



(b) 벡터방식

LX-800 프린터에서 출력시킨 폰트들



하고 쓰여질 크기를 정 의한다.

Font : DefaultFont =0;
TriplexFont =1;
SmallFont =2;
SansSerifFont =3;
GothicFont =4;
TriplexScriptFont =5;
SimplexFont =6;
ScriptFont =7;
ComplexFont =8;
EuroStyleFont =9;
BoldFont =10;

Direction : HorizDir =0…가로쓰기
VertDir =1…가로쓰기

CharSize : 글씨 크기는 1에서 최대 10까지이며 보통의 경우 1로 세팅된다.

● **OutText(TextString : string)**

문자열을 화면의 현재 포인터에 출력시킨다.

TextString : 출력할 문자열

〔리스트 3〕 예제 프로그램

```
program SampleFont;

uses Graph,   { library of graphics routines }
     Fonts;   { all the BGI fonts }
var
  GraphDriver, GraphMode, Error : integer;

procedure Abort(Msg : string);
begin
  Writeln(Msg, ': ', GraphErrorMsg(GraphResult));
  Halt(1);
end;

begin
  { Register all the fonts }
  if RegisterBGIfont(@TriplexScriptFontProc) < 0 then
    Abort('TriplexScript');
  if RegisterBGIfont(@SimplexFontProc) < 0 then
    Abort('Simplex');
  if RegisterBGIfont(@ScriptFontProc) < 0 then
    Abort('Script');
  if RegisterBGIfont(@ComplexFontProc) < 0 then
    Abort('Complex');
  if RegisterBGIfont(@EuroStyleFontProc) < 0 then
    Abort('EuroStyle');
  if RegisterBGIfont(@BoldFontProc) < 0 then
    Abort('Bold');
  if RegisterBGIfont(@GothicFontProc) < 0 then
    Abort('Gothic');
  if RegisterBGIfont(@SansSerifFontProc) < 0 then
    Abort('SansSerif');
  if RegisterBGIfont(@SmallFontProc) < 0 then
    Abort('Small');
  if RegisterBGIfont(@TriplexFontProc) < 0 then
    Abort('Triplex');

  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');  ←── 그래픽 활성화
  if GraphResult <> grOk then       ←────── 에러 체크
  begin
    Writeln('Graphics init error: ', GraphErrorMsg
    Halt(1);                          (GraphDriver));
  end;

  SetTextJustify(LeftText,CenterText);
  SetTextStyle(0,HorizDir,1);
  OutTextXY(50,10,'Default Font');
  SetTextStyle(1,HorizDir,3);
```

```
OutTextXY(50,30,'Triplex Font');
SetTextStyle(2,HorizDir,8);
OutTextXY(50,60,'Small Font');
SetTextStyle(3,HorizDir,4);
OutTextXY(50,90,'SanSerif Font');
SetTextStyle(4,HorizDir,4);
OutTextXY(50,120,'Gothic Font');
SetTextStyle(5,HorizDir,4);
OutTextXY(50,150,'TriplexScript Font');
SetTextStyle(6,HorizDir,4);
OutTextXY(50,180,'Simplex Font');
SetTextStyle(7,HorizDir,4);
OutTextXY(50,210,'Script Font');
SetTextStyle(8,HorizDir,4);
  OutTextXY(50,240,'Complex Font');
  SetTextStyle(9,HorizDir,2);
  OutTextXY(50,270,'EuroStyle Font');
  SetTextStyle(10,HorizDir,4);
  OutTextXY(50,310,'Bold Font');

  SetTextStyle(5,VertDir,5);
  OutTextXY(500,200,'Persnal');
  SetTextStyle(5,VertDir,5);
  OutTextXY(600,200,'Computer');

  readln;
  CloseGraph;
end.
```

● **OutTextXY(X, Y : integer ; TextString : string)**

문자열을 지정한 좌표에 출력시킨다. OutTextXY나 OutText 는 SetTextJustify 함수에 의해 설정된다.

● **TextHeight(TextString : string)**

TextString의 높이를 픽셀단위로 나타낸다. 이것은 그래프나 박스안의 표제 크기를 결정하는데 유용하다.

● **TextWidth(TextString : string)**

TextString의 폭을 픽셀단위로 나타낸다.

맺음말

이상으로 미약하나마 좁은 실력으로 몇자 적어보았다. 필자는 그래픽과 한글에 관심이 있어 이번 파스칼에서 사용되는 10개의 폰트와 한글폰트를 이용하여 보다 좋은 프로그램을 한번 짜보고 싶다. 아무쪼록 10개의 폰트를 이용하여 여러 훌륭한 프로그램들이 나왔으면 한다.

끝으로 이 원고에 도움을 주신 오승진 선배님께 감사의 말을 전하고 싶다.

마이크로프로세서 마이크로컴퓨터 講義案內

(하드웨어 과정)

개강 매월 5일 (2개월 완성)

본원에서는 금번 2개월 과정으로 digital circuit 설계로 부터 CPU 및 memory, 주변장치 등의 구조 및 동작원리를 습득하여 computer system 및 주변기기의 Interface를 설계하는 능력을 함양하고자 합니다. 또 이 과정에서는 Z-80계 computer 및 APPLE, IBM PC 등의 system을 대상으로 구성, 동작원리, 프로그래밍 실습을 하여 궁극적으로 MICOM 및 digital system의 설계 및 수리능력을 배양하고자 하오니 많은 참여 바랍니다.

■ 강습일정

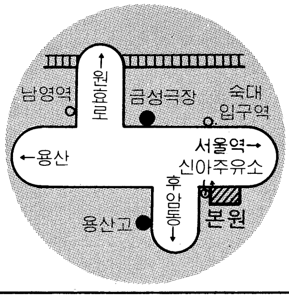
주별	일차	강습내용 (이론및실기)
1주	1일차	computer의 기본 구조와 8bit CPU의 종류 및 내부 구성
	2일차	논리회로 설계원리 및 그 응용
	3일차	마이크로 컴퓨터 시스템의 구성
2주	1일차	Karnaugh Map을 이용한 디지탈회로 설계
	2일차	Flip Flop 및 Counter 등을 이용한 응용회로 설계
	3일차	CPU의 주변회로 및 Address Map/Decoding
3주	1일차	TTL 논리회로 설계 실습
	2일차	A/D 및 D/A Converter 및 ALU
	3일차	Memory(ROM/RAM/DRAM)의 구성
4주	1일차	8080 및 8085 CPU의 구조 및 Timing 특성
	2일차	8085 Assembly언어 프로그래밍
	3일차	I/O(Port, Keyboard etc)의 원리 및 Video Generator

주별	일차	강습내용 (이론및실기)
5주	1일차	Z80 CPU의 구조 및 Timing 특성
	2일차	Z80 CPU의 Interrupt (IM0, IM1, IM2)
	3일차	μ-Computer의 회로설계 및 실습
6주	1일차	Assembly Language를 이용한 프로그램 실습
	2일차	8255 및 Z80 P10의 특성 및 응용
	3일차	16bit MPU(MC68000)의 구조 및 원리
7주	1일차	Z80 S10의 특성 및 응용
	2일차	EPROM Writer 및 Floppy Disk Controller
	3일차	MC68000의 Signal 및 Bus Operation
8주	1일차	음성합성 장치의 설계
	2일차	Bus Standard, Modem, Data통신 소개
	3일차	16bit MPU(MC68000)의 회로 설계

Textbook

1. TTL응용실무
2. Z80 마이크로컴퓨터
3. Microprocessor Interfacing Technique
4. Microprocessor and Digital Systems

Hardware + Software = ?



전자산업 4반세기를 지켜온 기술인의 전당

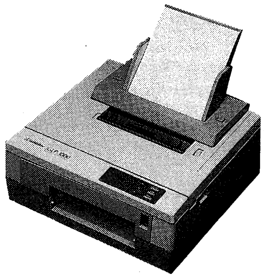
관인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

신제품

고속 레이저프린터
GLP-1000



GLP-1000은 EP카트리지에서 드럼과 토너부분을 분리, 교환할 수 있도록 하여 제품의 수명을 연장하고 유지비를 최소화시킨 전자사진방식의 레이저 프린터이다.

이 제품의 특징은 52dB 이하의 저소음과 분당 10페이지의 인쇄속도를 가지며 패러럴 포트와 시리얼 포트를 각각 독립적으로 사용할 수 있어 2대의 PC를 동시에 연결하여 사용할 수 있다. 또한 원, 타원, 사각형, 부채꼴 등의 도형과 도형내부에 여러가지 무늬의 페인팅이 가능하도록 그래픽 기능을 강화시켰다.

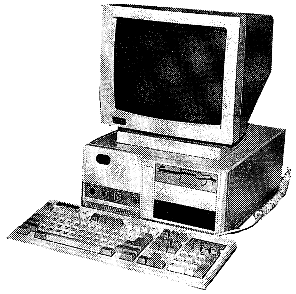
도트매트릭스 프린터를 대체해서 사용할 수 있으며 HP LASERJET+, DIABLO630, EPSON FX-80과 에뮬레이션이 가능할 뿐만 아니라 바코드 인쇄도 가능하다. 350mm×310mm 크기의 콘트롤러 보드를 2장 내장할 수 있어 사용자가 요구하는 추가기능을 사용할 수 있으며 한글은 KS 완성형 코드를 지원하며 한자는 행정전산망용 4,888자, 신문협의회의 5,170자를 지원한다.

제품의 규격은 다음과 같다.

- 인자방식 : 반도체 레이저 전자사진 방식
- 인자속도 : 10PPM
- 해상도 : 300DPI
- 수명 : 60만장 이상
- 소음 대기시 : 45dB 이하
  동작시 : 52dB 이하
- 듀티사이클 : 월 3,000~5,000장
- 최초인쇄 : 15초 이하
- 예열시간 : 75초 이하
- 내장폰트 한글명조 : 40×40, 48×48
  한글고딕 : 40×40, 48×48
- 인터페이스 : 센트로닉스 패러렐, RS-232C, 20mA Current Loop.
- 기타기능 : 문자회전, 강조, 문자기울기, 문자방향 조정

문의처 : 금성사
전 화 : (02)701-1114

21MHz의 초고속 PC
Beltron AT



이 제품은 I80286을 CPU로 채택하였고 페이지 모드 및 인터리브기법과 새도우 RAM을 사용하여 BIOS가 수행될 때 ROM의 대기 상태를 줄여 처리속도를 21MHz까지 향상시켰다.

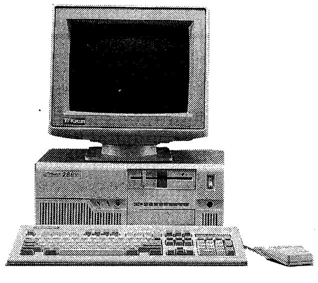
메인 메모리는 1MB가 기본이며 EMS메모리의 지원으로 최대 18MB까지 확장할 수 있다. 비디오 보드로는 CGA, MDA, HGA의 기능을 보유한 멀티비디오카드를 채택하였고 101키의 키보드와 처리속도 28m/s인 Miniscrible Intelligent 타입의 40MB HDD가 제공된다.

제품의 규격은 다음과 같다.

- CPU : I80286
- 메인메모리 : 1MB, 최대 16MB
- 보조기억장치 :
  FDD : 1.25MB(5.25인치)
  HDD : 40MB(5.25인치)
- 시리얼 포트 : 2개
- 패러럴 포트 : 2개
- 확장슬롯 : 8개
- 비디오보드 : 멀티 비디오카드
- 한글카드 : 옵션

문의처 : 마이크로텍스 엔지니어링
전 화 : (02)542-0960

16MHz/EMS 기능의 PC
TG-286V



TG-286V는 I80286 마이크로 프로세서와 80287 코프로세서를 탑재하였고 CPU보드와 멀티펑션기능을 하나로 집

약한 통합 마더보드로 설계되어 16MHz의 처리속도를 가진다. 또한 8MHz의 속도로 절환할 수 있어 I8088계열과도 호환이 가능하다.

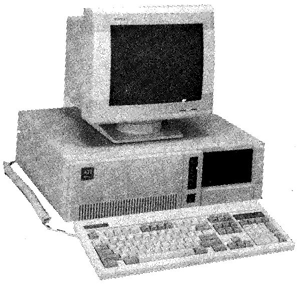
이 제품은 EMS기능의 보유로 별도의 확장보드없이 2MB까지 램메모리를 사용할 수 있고 시리얼 포트가 2개 있어 주변기기와의 연결이 용이하다. 또한 조합형/완성형 한글, 한자코드를 지원하며 4,888자의 한자가 제공되어 한글 인텔리전트 터미널로도 이용할 수 있다.

제품의 규격은 다음과 같다.

- CPU : I80286
- 출력속도 : 8/16MHz
- 메인메모리 : RAM : 1MB 최대 16MB
- 보조기억장치 FDD : 1.2MB
  HDD : 20MB
- 확장슬롯 : 8개
- 인터페이스 패러럴 포트 : 1개
  시리얼 포트 : 2개

문의처 : 삼보컴퓨터
전 화 : (02)390-2114

OA용 컴퓨터
ROMAX-ATT24



이 제품은 I80286-12 마이크로 프로세서를 CPU로 채택하였고 정격 DRAM (100n/s)을 사용하여 처리속도를 향상시켰다.

또한 MS-DOS 및 XENIX를 탑재하여 기억장치를 공유할 수 있어 멀티태스킹도 가능하다. IBM PC/XT/AT에 사용되는 각종 주변기기와 접속이 가능하며 리얼타임과 배치 프로세싱 등 대형의 단말기로 적합하며 AT끼리의 온라인이 이루어져 포인트 투 포인트와 클러스터방식을 지원한다.

CLOCK/CALENDAR 기능으로 절원시도 시간이 체크되며 KEY LOCK 장치가 있어 외부의 부주의로 인한 프로그램과 데이터가 파괴되는 것을 안전하게 방지할 수 있다.

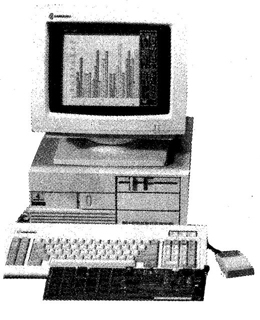
한글은 KSC-5601 2바이트 완성형 코드체계를 채택하여 명조체의 한글 2,350자와 4,888자의 한자 및 일본어, 특수부호들의 사용이 가능하며, 그래픽 모드(640×400)에서 한글, 한자사용이 가능하다.

제품의 규격은 다음과 같다.

- CPU : I80286
- 메인메모리: RAM : 1MB/최대 15MB
  ROM : 64kB
- 보조기억장치: FDD : 1.2MB
  (5.25인치)
  HDD : 20MB
- 인터페이스:패러럴포트 : 2개
  시리얼 포트 : 2개
- 확장슬롯 : 8개

문의처 : (주)로얄컴퓨터
전 화 : (02)785-3458

PC 팩시밀리
SFA-3000



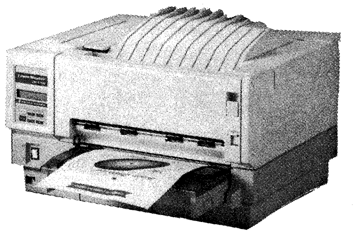
SFA-3000은 어댑터 카드와 통신 소프트웨어로 구성되어 워드프로세서로 작성된 문서와 데이터를 상대의 팩시밀리나 PC로 전송은 물론 수신까지 할 수 있는 제품이다. 또한 PC가 직접 상대방의 팩시밀리와 통신하므로 찌그러짐이나 번짐이 없는 선명한 화상을 전송할 수 있고 수신된 화상은 편집할 수도 있다.

제품의 주요기능은 다음과 같다.

- 동보송신 : 최대 200건
- 예약송신 : 송신개시 시각지정
- 수신처, 발신처, 발신시각의 자동기입기능
- 수신처 선택기능
- 송신대기중인 수신처 관리기능
- 자동 재송신기능
- 송신모드 : 표준, 파인, 축소모드
- 수신기능

문의처 : 삼성전자
전 화 : (02)559-2644

레이저 프린터
LBP-X1100



이 제품은 프린터 조작, 표시 판넬, 페이퍼카세트 등을 사용하기 쉽게 전면에 배치하였으며, 토너카트리지 및 드럼 카트리지교환 시점을 판넬상에 표시해 줌으로써 프린터 조작을 쉽게 했다.

또한 전자 사진부를 완전히 카트리지화하여 소모품인 토너와 드럼을 분리할 수 있어 사용자가 손쉽게 유지, 보수는 물론 간단하게 교환할 수 있게 하였다.

도트 매트릭스 프린트와도 호환성이 가능하여 IBM PC S/W의 출력기로써 응용이 가능하며, 소형컴퓨터, 미니컴퓨터 등과도 접속할 수 있다.

제품의 규격은 다음과 같다.

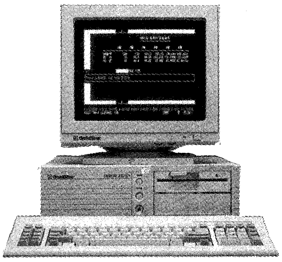
- 프린트방식 : LASER. XEROGRAPHY방식
- 형식 : 탁상용 페이지프린터
- 해상도 : 240dpi

●PC레이다●

• 프린트속도 : 11PPM/9PPM
• 사용문자 : 한글(명조, 고딕), 한자(명조), 영문
• 용지크기 : 최대 B4
• 인터페이스 : 패러렐
• 문자구성 : 32×32

문의처 : 제철엔지니어링
전 화 : (02)733-3551

5MB까지 확장가능한 PC
마이티AT/3$^+$



이 제품은 I80286을 CPU로 채택한 16MHz의 클럭속도를 가지는 16비트 컴퓨터로 IBM PC/AT와 호환이 가능하다.

이 제품의 특징은 1MB의 기본 기억용량을 별도의 메모리보드없이 5MB까지 확장시킬 수 있으며 사무처리 및 각종 CAD/CAM 작업에 적합하다.

제품의 규격은 다음과 같다.

• CPU : I80286
• 클럭속도 : 16MHz
• 메인메모리 RAM : 1MB/최대 5MB
• 보조기억장치 FDD : 1.2MB
  HDD : 64MB
• 시리얼 포트 : 1개
• 패러렐 포트 : 1개

문의처 : 금성사
전 화 : (02)701-1114

드럼타입 플로터
GRX-400

이 제품은 펜 교환횟수를 최소한으로 줄여 드로잉을 빠르게 하는 펜 최적화(Pen Optimizer) 기능을 가지고 있으



며 플로터 파라미터 세팅을 A4에서 A0의 크기까지 다양하게 할 수 있는 플로터이다.

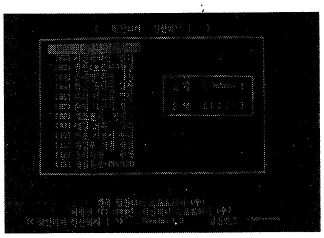
또한 대형 LCD 디스플레이를 장착하여 변수설정이 용이하며, HP-GL 명령어와 호환되는 RD-GLDII 명령어가 내장되어 대부분의 CAD, CAM 소프트웨어를 활용할 수 있다.

제품의 규격은 다음과 같다.

• 최대 플로팅영역 : 1155×831mm
• 해상도 : 0.025mm/스텝
• 용지크기 : A0~A4
• 펜의 종류 : 잉크펜, 세라믹펜, 볼포인트펜, 파이버팁펜
• 펜의 수 : 8개
• 플로팅 속도 : 60mm/s
• 버퍼의 크기 : 18kB

문의처 : (주)태일시스템
전 화 : (02)756-5340

회계패키지
팔란티어 전산회계(I)



팔란티어 전산회계(I)은 전표처리가 용이하고 시산표 출력만으로 작업이 자동으로 처리되는 회계패키지이다.

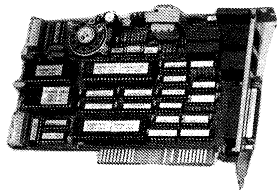
이 제품은 사용법이 간단하고 특히 회사 및 업무규모, 처리방식에 따라 적절한 프로그램을 선택할 수 있다.

사용환경 조건은 다음과 같다.

• 사용기종 : IBM PC XT/AT
• 한글처리 : KSC5601 표준코드, 2바이트 조합형코드
• 주기억용량 : 256kB
• 보조기억용량 : HDD 10MB 이상
• O/S : MS-DOS 한글 MS-DOS 2.0
• 출력가능한 자료 : 일계표, 월계표/시산표/원장/총계정원장/제조원가명세/결산 재무제표/결산 시산표/결산 대차대조표/결산 손익계산서/결산 제조원가명세

문의처 : 한국 팔란티어
전 화 : (02)785-0597

PC 카드 모뎀
GTM-1212 PC2



이 제품은 내장형 모뎀으로 체신부 형식승인을 획득한 BELL 규격과 CC-ITT의 권고안에 의해 설계되었다.

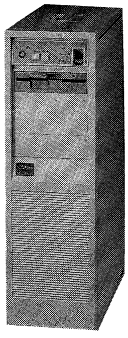
이 제품의 특징은 동기 및 비동기방식으로 설계되었고 시리얼카드 기능을 겸비하여 데이터 전송효율의 증대 및 부대정비설치 비용절감과 신뢰도를 향상시킨 것이다.

제품의 주요규격은 다음과 같다.

• 적용회선 : 공중회선용 2선식 전이중방식
• 데이터 전송율 :
  동 기 식 : 1200BPS±0.01%
  비동기식 : 1200BPS+1%, -2.5%
• 송신출력레벨 : -10dBm
• 수신가능레벨 : -43dBm
• 시험기능 : 아날로그, 디지탈, 원격디지탈 루프
• 운용방식 : 동기 및 비동기식

문의처 : (주)자네트 시스템
전 화 : (02)546-4741

32비트 엔지니어링 워크스테이션
EXP386



EXP386은 IBM PC/XT/AT와 그 호환기종의 모든 소프트웨어를 그대로 사용할 수 있는 클럭속도가 25MHz인 32비트 워크스테이션이다.

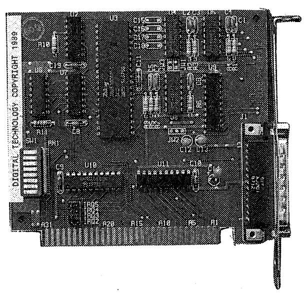
MS-DOS, UNIX, OS/2를 O/S로 사용할 수 있고 FA, CAD/CAM/ CAE 분야와 시뮬레이션분야에 이용이 가능하다.

제품의 규격은 다음과 같다.

- CPU : I80386
- 클럭속도 : 25MHz
- 메인메모리 : RAM : 4MB
- 보조기억장치 FDD : 1.2MB
  HDD : 64MB
- O/S : MS-DOS, UNIX V.3, UNIX SYSTEM, XENIX 286/386

문의처 : 제우스컴퓨터
전　화 : (02)784-7841

XT, AT용 통신카드
DT-LSL8



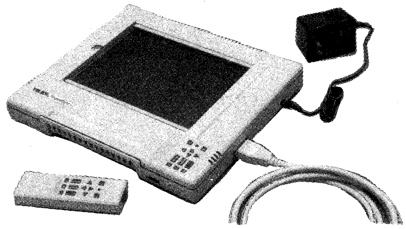
이 통신카드는 MS-DOS상에서 동작되며 IBM XT/AT 및 호환기종 또는 386 등에서 사용할 수 있는 통신카드이다.

이 제품은 스타랜(STAR LAN)방식으로 각 시스템을 연결하여 초보자도 이용하기 쉽도록 운영방식이 메뉴방식으로 되어 있다. 또한 운용 소프트웨어인 DT.COM은 램상주 프로그램으로 되어 있어 다른 작업중에서도 통신이나 화일전송이 필요시 언제든지 통신이 되도록 되어 있으며 RS-232C 통신을 채택하여 접속시스템에서 RS-232C 포트가 있다면 별도의 통신카드를 요구하지 않는다는 것이 특징이다.

- 전송속도 : 38.4kbps
- 제한거리 : 30m
- 사용 가능한 시스템수 : 4, 8대

문의처 : 디지탈테크놀러지
전　화 : (02)581-1511

한글처리가 가능한 영상장비
MagnaByte 시리즈



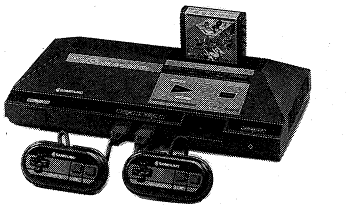
한글처리 문제를 해결한 컴퓨터 영상장비인 이 제품은 컴퓨터 모니터의 신호를 LCD판으로 받아 오버헤드 프로젝터를 통해 스크린에 투사하는 장비로서 각종 회의나 브리핑, 교육 등에 컴퓨터를 리얼타임으로 이용할 수 있어 용도가 다양하다.

그리고 이 제품은 컴퓨터, 비디오 해상도를 자동으로 인식, 보정해주는 기능을 내장하고 있고, EGA, CGA는 물론 허큘리스, MDA, MacII, 매킨토시 512/+/SE 및 VGA의 전기종과 한글, 영문, 그래픽의 전모드에서 동작한다.

또한 16가지의 칼라 및 패턴을 사용자가 임의로 선택할 수 있으며 다양한 기능의 리모콘과 LCD 판 자체에서 제공하는 헬프 스크린, 브리핑 전용의 강력한 슬라이드 소프트웨어 등을 함께 제공하고 있다.

문의처 : 삼아상사
전　화 : (02)730-9438

게임용 홈컴퓨터
SPC-100



이 제품은 가정용 TV에 부착하여 사용할 수 있는 홈컴퓨터로 다양한 게임용 S/W(팩)과 조이패드를 사용하여 1인 또는 2인이 동시에 게임을 할 수 있는 가정용 게임컴퓨터이다.

또한 64가지 색상을 지원하고 풍부한 사운드를 제공한다.

제품의 규격은 다음과 같다.

- CPU : Z80(4MHz)
- ROM : 8kB(내장)
- RAM : 8kB(내장)
- V-RAM : 16kB(내장)
- 그래픽표시 : 256×192도트
- 색상 : 64색
- 패턴 : 8×8도트
- 스크롤 : 상, 하, 좌, 우, 대각선 부분이동
- 사운드 : 3중화음+1 노이즈
- 슬롯 : 카드용 35P, 카트리지용 44P 확장단자용 50P
- 조이스틱 : 2개(분리가능)

문의처 : 삼성전자
전　화 : (02)559-2616

신간안내

MIS원론

소영일외 1인 편저/법영사 발행/
정가 9,000원/675면/4·6배판



격변하는 국내외 환경에 효과적으로 적응하기 위하여 정보시스템에 관한 충분한 지식이 절실히 요구된다. 따라서 정보 시스템에 관한 이해에 도움을 줄 전문서적이 필요하게 되었다.

이 책은 이에 부흥하기 위하여 사회계열의 학생들과, 기업에서 경영자가 되고자하는 사람 등 정보시스템에 대한 포괄적인 지식을 얻고 싶어하는 사람들을 주요대상으로 쓰여졌다. 따라서 이 책에서는 컴퓨터의 기술적인 측면보다는 컴퓨터를 활용하는 경영자 측면을 강조하여 정보시스템의 기본원리, 개념, 아이디어의 이해를 돕고 있다.

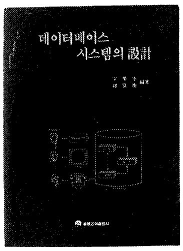
이 책의 구성은 3편으로 나누어져 있으며 1편에서는 조직의 관리와 통제문제를 기술하고 있으며 또한 경영정보 시스템의 본질에 관하여 다루고 있다.

2편에서는 컴퓨터 시스템의 기초와 컴퓨터 산업에 관하여 기술하였다.

3편에서는 경영자의 입장에서 시스템 프로젝트를 어떻게 편성하고 수행할 것인가 하는 것과 또 필요한 컴퓨터를 어떻게 선택할 것인가 하는 것을 다루고 있다.

데이터베이스 시스템의 설계

송영재외 1인 편저/
홍릉과학출판사 발행/
정가 6,500원/370면/4·6배판

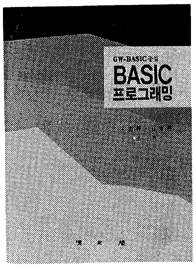


데이터베이스라는 개념이 컴퓨터 세계에 등장한 이래 물질에너지와 함께 정보가 귀중한 자원이라는 생각이 확산되고 정착되어가고 있는 추세이다. 또한 작업처리 중심에서부터 데이터 중심으로 전환되면서 대량으로 생산된 정보가 사용자측의 요구에 맞추어 적시에 액세스할 수 있는 단계로 진전되었다. 이에 부흥하여 많은 데이터베이스에 관한 서적이 소개되었으나, 너무 이론적인 것에 치중한 나머지 실질적으로 데이터베이스를 설계하는 기법이 부족하였다. 그러나 이 책은 데이터베이스의 개념을 많은 그림을 통하여 이해에 역점을 두고 설명하고 있다.

또한 데이터베이스의 어프로치, 데이터베이스 시스템의 구성, 계획, 모델화, 정보생산 시스템의 설계, 논리·물리 데이터 모델의 설계를 섬세하게 다루고 있다.

BASIC 프로그래밍

송도선·최동선 공저/청문각 발행/
정가 5,500원/254면/4·6배판



컴퓨터 보급댓수에 비하여 컴퓨터를 제대로 이용할 줄 모르는 사람이 아직도 많은 실정이다. 그래서 이 책은 최근 가장 많이 쓰이고 널리 이용되는 16비트 컴퓨터의 GW-BASIC을 사용하기 위한 초보자용 지침서이다.

이 책은 일반적인 컴퓨터 개론과 MOS-DOS의 기본사용법 그리고 베이직의 기본과 GW-BASIC의 명령어 등을 설명하고 있다. 또한 그래픽과 사운드 등을 이론과 예제를 함께 실어 실행과 동시에 숙달되도록 하였다. 그리고 응용과 실무부분에서는 화일의 사용과 응용을 실제적으로 사용할 수 있게 하였다.

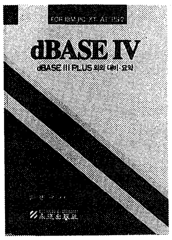
이 책은 전체 6장으로 구성되어 있다. 1장과 2장에서는 컴퓨터의 일반적인 개요와 MS-DOS에 대하여 설명하여 초보자들의 이해를 돕고 있으며 3장과 4장에서는 베이직언어에 대하여 설명하고 베이직 명령어를 예제를 통하여 설명하고 있다. 5장과 6장에서는 응용프로그램과 실무프로그램을 예제로 기재하여 초보자 학습에 효과를 배가하고 있다.

부록에서는 에러메세지와 ASCII 코드 등 베이직의 학습과 직접적인 현관은 적지만 컴퓨터를 이용하는데 유용하게 사용될 자료를 게재하였다.

---

dBASE Ⅳ

김경태 편저/영진출판사 발행/
정가 4,000원/123면/4·6배판



dBASEⅣ에 대한 전반적인 소개와 그 구성을 설명한 이 책은 특히 현재 dBASEⅢ+ 사용자가 dBASEⅣ와 dBASEⅢ+를 대비 판단을 가늠할 수 있게 기술하였다.

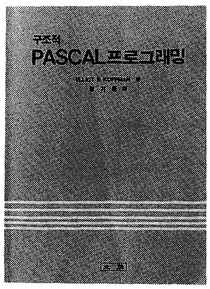
그리고 dBASEⅢ+ 사용자가 이 책을 이용하여 어느 부분이 향상되고 보강된 것인지 판단할 수 있도록 배려하였다.

이 책의 내용은 8개의 장으로 구성되어 있는데 1장과 2장에서는 dBASEⅣ의 개요와 인스톨, 데이터 처리의 강력성, 호환성, 확장성에 대하여 설명하고 있다. 3장부터 6장까지는 dBASEⅣ의 사용방법에 대하여 체계적으로 설명하며, 에디터의 사용에 대해서도 기술하였다.

7장부터 8장까지는 dBASEⅣ의 메뉴계통도를 그림을 통하여 설명하고 있고, 명령어, 함수에 대해서 기술하고 있다.

구조적 PASCAL 프로그래밍

정기동 역/생능출판사 발행/
정가 8,000원/659면/4·6배판



이 책은 파스칼 프로그래밍을 세심하게 그리고 조직적으로 학습할 수 있는 입문과정용 학습서이다.

그리고 이 책의 주된 목적은 파스칼 사용자들에게 알고리즘을 단계적으로 개발해 나가는 것과 이것을 통한 문제해결 능력을 배양하는데 있다. 또한 프로시듀어 개념을 숙지시키기 위하여 프로시듀어를 사용한 실질적인 프로그래밍을 소개하여 숙지시키고 있다.

각절의 끝에는 연습문제를 게재하여 학습한 내용을 스스로 점검하고 평가할 수 있게 하였다.

시스템 분석과 설계

이영환 저/법영사 발행/
정가 8,000원/434면/4·6배판



이 책은 성공적인 MIS의 구축 및 활용을 위하여 쓰여진 시스템 설계 입문서이다.

시스템의 기본개념을 이해시키는데 많은 비중을 두고, 사례연구를 통하여 개념의 중요성을 환기시키고 있다. 또한 혼돈하기 쉬운 시스템 분석가와 프로그래머의 업무차이를 설명함으로써 시스템 분석가의 역할을 이해시키고 있다.

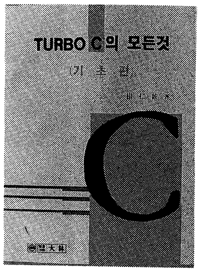
또한 MIS(경영정보시스템)의 이용자적측면과 기능적 측면을 설명함으로써 MIS의 개념을 혼란없이 정립시키고 있다.

시스템 개발주기를 그 순서에 따라 차례로 설명해 나가면서 각 단계별 업무와 사용되는 도구는 무엇인가를 매끄럽게 설명하고 있다.

이 책은 시스템 분석가를 지망하는 학생이나 프로그래머 또는 경영자들이 읽어 볼만한 서적이다.

TURBO C의 모든 것(기초편)

최인현 저/대림출판사 발행/
정가 5,500원/297면/4·6배판



현재 PC용 C언어의 주류는 터보C와 MS-C로 대별되고 있으며 각 C언어의 버전들은 나름대로 특성을 가지고 있다. C언어의 생명은 속도라고 할 수 있는데 터보 C는 그 속도가 기계어에 가까울 정도로 빠르고 소프트웨어가격도 저렴하여 사용자들의 관심을 끌고 있다.

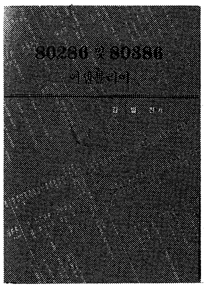
본서에서는 컴퓨터 프로그래머나 초보자들에게 C언어로의 접근에 있어 자신감을 주기 위하여 터보 C의 특성을 쉽게 설명하고 있다.

본서는 예제를 통한 실제적인 내용을 기술하여 독자들의 이해를 돕고 있으며 후반부에서는 터보 C 버전 1.5를 이용한 그래픽의 기초에 관하여도 설명하고 있다.

최근 몇년사이에 급격하게 보급된 터보 C의 유저확산과 더불어 초보자들에게 적당한 안내서가 부족했던 현시점에서 본서는 초보자들의 좋은 안내서가 될 것이다.

80286 및 80386 어셈블리

김병천 저/자유아카데미 발행/
정가 6,000원/285면/4·6배판



어셈블리어와 컴퓨터 구조에 관한 지식은 이들의 새로운 기술과 지식이 계속 발표되고 있기 때문에 지속적인 학습이 있어야 좋은 소프트웨어를 개발할 수 있다.

본서는 80286과 80386의 중앙처리장치에 관하여 설명하고 있는데 하드웨어적인 관점보다 소프트웨어적인 면에 중점을 두고 있다.

특히 어셈블리어를 통해 소프트웨어를 작성할 때 필수적인 인터럽트에 대한 설명을 자세히 다루었으며, 어셈블리어와 C언어의 링크를 설명하면서 타 언어와의 링크도 언급하고 있다.

또한 수학적 함수의 응용을 예로 들어 수치보조장치에 관한 설명과 데이터의 전송형태와 통신콘트롤러, 레지스터, 통신 프로그램의 작성 등 데이터통신에 관한 설명을 추가하여 시스템 엔지니어의 지침서가 되도록 하였다. PC

未進出版社 새로 나온 책

정상의 프로그래머들이 선택하는 화제의 책!!



초보자를 위한
MS-DOS 入門

김 우용 저 4×6배판 값 5,000원

컴퓨터에 흥미를 느껴 이제 공부를 시작하려는 분, 처음으로 컴퓨터를 구입해서 사용하려는 분들에게 있어 가장 쉽고 가장 훌륭한 지도교수가 될 DOS 입문서. 본문에서는 DOS의 기초와 활용, 효율적인 화일 관리, BATCH 화일의 작성과 이용, 하드디스크 사용법 등의 내용을 그림과 화면을 곁들여 쉽게 설명하였고, 부록에는 전반적인 DOS 명령어를 수록하여 보다 자연스럽게 MS-DOS에 접근할 수 있도록 하였다. IBM PC를 처음 사용하게 될 때 필요한 기초지식에서부터 운영체제인 MS-DOS 활용법에 이르기까지 초보자들이 단계적으로 학습하여 활용할 수 있도록 비유를 통해 쉽게 설명하여 독학생에게는 물론 컴퓨터 메이커의 매뉴얼, 학원 및 학교, 회사의 교재로도 적합하도록 엮었다.

고급 사용자를 위한 (제Ⅱ집)
실무 Lotus 1-2-3

홍 수현 편저 260×380㎜ 값 12,000원



Lotus 1-2-3의 꽃이라 할 수 있는 매크로와 함수의 메뉴 선택법을 I집보다 한 단계 높여 작성함으로써 보다 편리하고 능률적으로 업무에 이용할 수 있도록 엮었다. 재무회계 관리, 제품출고 관리, D.M 발송 관리 및 어음 관리 등의 내용을 자세히 설명하고 있으며, 또한 사용자들의 편의를 위해서 이 내용을 디스크에 수록하여 업무에 즉시 활용할 수 있도록 공급하고 있다 (Software 4장 값 40,000).

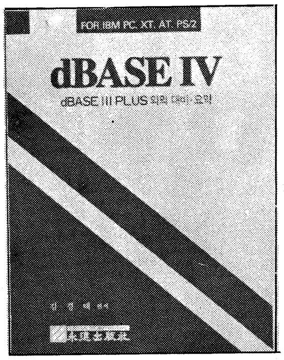
초보자를 위한 가이드 **IBM PC 入門**

탁 연상 저 4×6배판 값 4,000원



이 책은 IBM PC 사용자가 기본적으로 알아야 할 내용들을 하나씩 이해하기 쉽게 설명하고 있으며, 특히 컴퓨터를 처음 배우려는 초보자들에게 친절하고 자세한 안내서가 될 것이다.

IBM PC를 구입한 초보자가 당신이거나 혹은 그런 분을 알고 있다면 이 책보다 더 좋은 선물은 없을 것이다. 또한 컴퓨터 메이커의 매뉴얼이나 학원, 학교 교재로도 적합하게 만들었다.

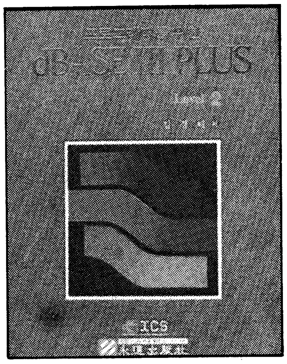


dBASE IV (dBASE III PLUS 와의 대비·요약)

김 경태 편저 4×6배판 값 4,000원

dBASE III plus에 익숙해져 있는 국내 user들에게 dBASE IV의 향상된 기능을 소개하고 그 개요를 설명함으로써 새로운 Version에 대한 이해를 돕고자 하였다.

● Single user와 Multi user용의 구분된 package ● Control Center에 의한 일괄처리 menu driven 방식 채택 ● Development edition package를 별도로 지원, 응용 program개발의 완벽한 tool을 제공 ● Window 기능, 내장 Compile기능, Graphic Software(CHART MASTER)와의 접속



프로그래머를 위한 dBASE III PLUS 2

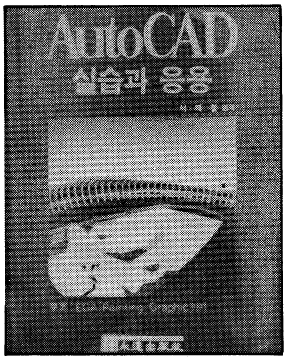
김 경태 저 4×6배판 값 6,000원

이 책은 dBASE Programmer에게 가장 친근한 지침서로, Database file을 Programming에서 효과적으로 사용할 수 있도록 작성하는 방법과 Indexing technique, 프로그램의 실체를 다루는 Command file의 기본 운영체제와 기능별, 사례별 Simulation module, 특히 60여개 이상의 단위 예제 프로그램은 Programmer로 하여금 그 응용성을 고려할 수 있도록 배려하였다. 또한, 곧 출간 예정인 Level 3에서는 실무 분야별 응용 프로그램을 완성편으로 정리하여 발행할 예정이다.

AutoCAD 실습과 응용

서 재철 편저 국배판 값 10,000원

이 책은 실무에서 AutoCAD를 사용하던 사람이 직접 쓴 책으로, Manual식의 피상적인 접근이 아닌 실전 실습과 응용으로서 높은 부가 가치를 창출할 수 있는 AutoCAD의 지침서이다. 특히 EGA PAINT라는 좋은 Graphic Program까지 실어서 AutoCAD의 미비점을 보강하여 Graphic 화면을 최대한 살릴 수 있게 하였으므로, 분명 이 책은 평면적이고 2차원적인 Draft가 아닌 입체적이고 환상적인 EGA Color의 세계를 열어줄 것이다.



◘ 건설 견적업무를 PC로 (dBASE III+ 공사내역서 작성 프로그램집)	김 재 율 저 4×6 배판 값 8,000원
◘ 단기완성을 위한 Lotus 1-2-3 300제	김우용·홍수현 편저 4×6 배판 값 6,000원
◘ PC 통신시리즈 II LAN 입문 (PC Communications & Networking)	묵현상·정영일 편저 4×6 배판 값 7,500원
◘ Microsoft QuickBASIC 4.0 활용 (Lavel I)	장 석 원 저 4×6 배판 값 5,000원

최첨단 도서 전문 출판
永進出版社
서울시 용산구 원효로 1가 42-17/302호
☎ 702-1119(代), 719-1119·1190
FAX 719-6550

●광장●

뭐든지 물어보세요

칼라TV를 모니터로 사용할 수는 없는지

본인은 EGA카드를 소유하고 있는데 칼라모니터의 가격 때문에 구입을 못하고 있습니다. 혹시 칼라TV와 연결할 수 있는 방법은 없을까요?

장상훈 / 서울시 마포구 망원동 316-6호

전기적 특성의 차이때문에 불가능합니다.

우선 칼라 디스플레이장치에 대해 설명을 드리는 것이 도움이 될 것 같습니다. 칼라 디스플레이카드에는 CGA, EGA, VGA, PGA 등이 있읍니다. 여기서 CGA만은 다른 것과 차이점이 있읍니다.

컴퓨터의 처리내용을 모니터에 표시하는 비디오부분은 일반적인 TV수상기와 처리하는 과정이 크게 다르지는 않습니다. 다만 모니터에 표시하는 프레임(Frame)과 주사방법에 차이점이 있습니다. TV에서는 수직주사 주파수를 60Hz로 수평주파수는 15750Hz로 비디오신호를 스크린에 재현시킵니다.
이와 동일한 기능을 갖는 것이 CGA 모니터 입니다. 따라서 CGA 카드를 꼽을 경우에는 바나나 잭을 사용하면 칼라 TV와 연결이 가능합니다.

그러나 디스플레이 카드장치에서 나오는 연결단자에 따라 콤포지트(Composite) 형과 세퍼레이트(Separate)형으로 구별되는데, 콤포지트형은 애플 컴퓨터 등 8비트 컴퓨터에서 볼 수 있는 연결단자로 위에서 언급한 수직신호와 수평신호, 비디오신호가 한개의 선으로 복합된 것을 말하며 세퍼레이트형은 IBM PC에서 볼수 있는 D쉘(Shell) 코넥터로서 각각의 신호가 분리되어 나오는 형태를 말합니다.

독자가 보유하고 있는 EGA카드의 수직주사 주파수는 대략 50~60Hz, 수평주사 주파수는 15750~21850Hz 정도일 것입니다. 각 주파수가 상당히 유동적이기 때문에 모니터가 이를 감당해야 합니다. 그러나 칼라 TV는 주사주파수가 고정되어 있어서 EGA카드를 연결하는 것은 전기적 특성의 차이때문에 현재로서는 불가능합니다.

터보스위치를 부착하려면 어떻게 하는지

저는 1년반전에 XT 컴퓨터를 구입해 사용하고 있읍니다. 그런데 제가 갖고 있는 컴퓨터에는 터보스위치와 리세트 스위치가 없읍니다. 혹시 제가 갖고 있는 PC에 이 두가지 스위치를 부착할 수 있는지요. 부착할 수 있다면 거기에 대한 자세한 배선도를 알려 주시면 고맙겠읍니다.

김성우 / 서울시 성북구 정릉4동 286-18 12 / 3

점퍼를 이용하면 간단히 부착이 가능합니다.

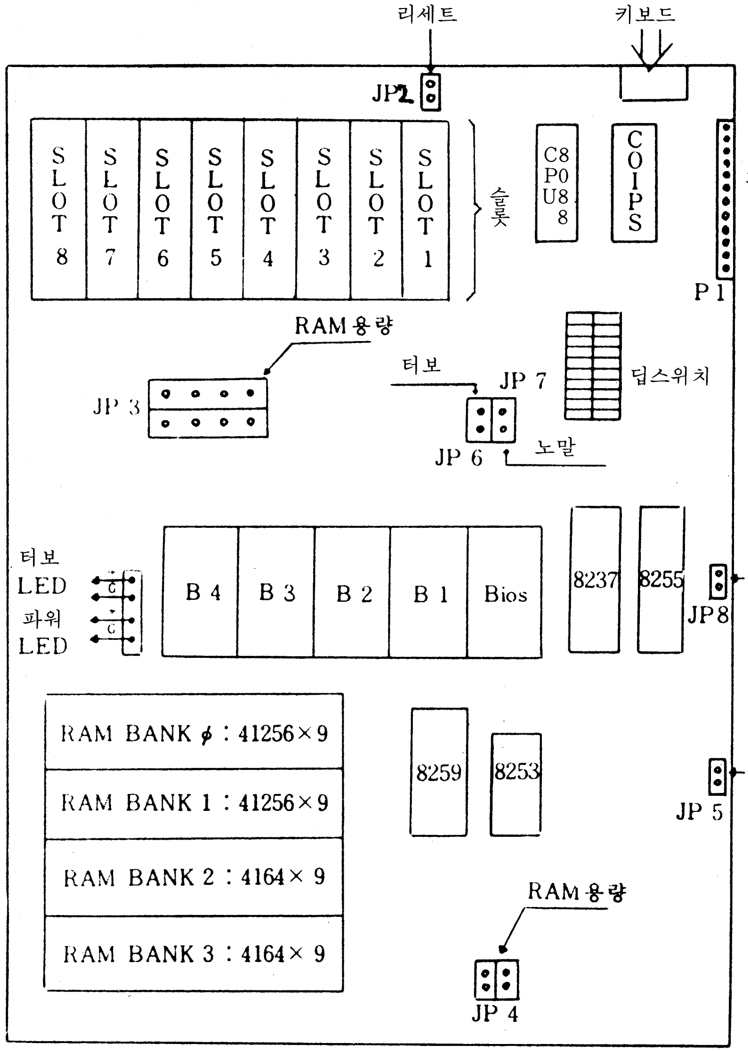
귀하께서 보유하고 계신 컴퓨터의 정확한 사양을 제시하지 않았기 때문에 명쾌한 답변은 어려우나 근접한 예를 들어 설명하겠읍니다.

우선 터보스위치는 원래 IBM 퍼스널컴퓨터 호환기종에서만 볼 수 있는 것으로 CPU의 동작클럭 주파수를 제어합니다. 스피드제어에는 하드웨어적으로 전환하는 방법과 소프트웨어적으로 전환하는 방법이 있으나 대부분의 호환기종들은 2가지 방법을 동일기계에서 병행하고 있습니다.

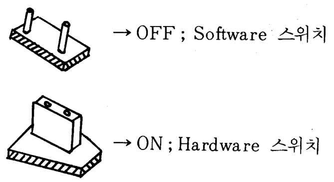
만약 컴퓨터의 기종이 IBM PC / XT 표준 보드라면 터보스위치는 부착할 수 없고 호환기종이라면 부착할 수 있읍니다. 이 터보스위치의 부착가능 여부는 컴퓨터가 부팅된 상태에서도 판단할 수 있읍니다. 즉 어느 때라도 부팅상태에서 Ctrl-Alt-(−)와 Ctrl-Alt-(+)키를 동시에 누르면 커서모양이 바뀌거나 본체에 있는 스피커에서 단음 또는 2중음이 발생합니다. 이러한 현상이 발생되면 터보스위치 부착이 가능하다는 뜻입니다.

이렇게 부착이 가능하다는 판단이 내려졌을 때에는 PC 구입시 받은 하드웨어 매뉴얼을 참조하여 보드상에서 점퍼 핀(Jumper Pin)을 찾읍니다. 이 점퍼핀에 코넥터를 연결하여 외부에 스위치를 만들어 접속하면 하드웨어 및 소프트웨어적으로 동작클럭 스피드를 제어하는 터보스위치가 되는 것입니다.

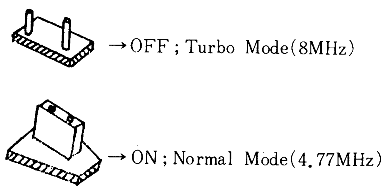
〔그림 1〕 XT호환기종의 보드



JP2 ; Reset pin
JP8 ; Key-Lock Switch
JP5 ; Turbo 스위치의 기능 선택(Software 또는 Hardware)



JP5 ; Turbo 스위치(Hardware)



JP6 ; Turbo LED단자
〈스위치 접속〉
Turbo 스위치→JP5에 접속
Reset 스위치→JP2에 접속

리세트 스위치도 보드상에 리세트핀이 준비되어 있읍니다. 이 핀의 위치는 매뉴얼을 참조하면 쉽게 찾을 수 있읍니다. 이 리세트핀에 코넥터를 연결하여 스위치를 부착하면 리세트스위치가 됩니다. 그러나 IBM PC/XT 표준보드라면 전기적인 지식을 갖고 있는 사람에게 문의를 하셔야 합니다.

좀더 이해를 돕기 위해 87년도에 나왔던 4.77/8MHz XT보드를 예로 들어 실제로 스위치를 부착하는 방법에 대해 알아 보겠읍니다.

우선 준비해야할 부품은 암코넥터 2핀짜리 3개와 록스위치(Lock Switch : Push Button-터보스위치용), 언록스위치(Unlock Switch ; Push Button-리세트 스위치용 ; 누르면 접점만 붙고 떼면 원위치로 돌아오는 스위치), 납땜인두, 납, 니퍼 등이며 이들은 청계천 세운상가의 1층이나 3층에 있는 노점상에서 쉽게 구입할 수 있읍니다. 코넥터를 구입할 때는 전선이 반드시 연결된 것을 구입하십시오. 그리고 코넥터와 스위치가 이미 연결되어 있다면 납땜은 할 필요가 없을 것입니다.

그러나 연결되지 않은 상태라면 코넥터에 연결된 전선끝의 피복을 약 2mm 정도만 벗겨내고 스위치의 접점위치를 확인하여 납땜을 한 다음, 스위치를 컴퓨터 본체나 외부의 적당한 장소에 고정하고 그림 1에 표시한 위치에 코넥터를 연결하면 됩니다.

터보스위치 동작여부는 LED를 연결하면 직접 확인할 수 있읍니다.

그리고 앞에서 설명한 작업을 할 때 컴퓨터의 전원을 OFF상태에서 해야하며, 전선이 벗겨져서 내부의 어떠한 부품과도 접속되지 않도록 항상 주의해야 합니다.

인공지능 언어란 기존의 언어와 어떻게 다른지

요즘 인공지능용 프로그래밍 언어라는 말을 자주 듣게 됩니다. 도대체 인공지능 언어는 어떤 것이며 기존의 컴퓨터 언어와는 어떤 차이가 있읍니까?

윤명상 / 충남 대천시 대천동 145-16

논리적으로 추론이 가능하다는 것이 큰 차이점

컴퓨터는 사용하는 소자에 따라 제1세대(진공관을 쓰던 시대), 제2세대(트랜지스터를 쓰던 시대), 제3세대(집적회로를 쓰던 시대)와 제4세대(초고밀도 집적회로를 쓰는 시대)로 구분할 수 있읍니다. 즉 지금은 4세대 컴퓨터 시대라고 볼 수 있는데 앞으로는 대규모의 지식정보를 통한 인공지능, 즉 생각하는 컴퓨터 구현(제5세대 컴퓨터)의 시대가 될 것이며 이미 각계에서는 이 프로젝트 실현에 많은 투자를 하고 있읍니다.

이 5세대 컴퓨터를 위한 프로그래밍 언어에도 많은 연구가 진행되어 왔으며 프롤로그(Prolog), 에이다(Ada), 리스프(Lisp), 플래너(Planner), KRL 등은 이미 우리들의 귀에도 익숙해져 있읍니다(특히 터보 프롤로그와 리스프언어인 경우는 현재 PC에서 많이 쓰이고 있음).

끝으로 이들 언어의 특징을 보면 기존의 언어가 가, 감, 승, 제 및 비교판단기능을 기본으로 수행하는데 비해 인공지능 언어는 대량의 지식정보를 구축할 수 있는 데이터 베이스기능과 수식이 아닌 스트링처리를 기본으로 한다는 점입니다. 그래서 프롤로그의 경우를 보면 일정한 규칙들과 여러가지 사실(현상)들이 주어지면 논리적으로 추론해 내는 것을 볼 수 있읍니다.

dBASE Ⅲ+에서 한글이 깨지지 않게 하려면

삼보트라이젬 88+를 사용중에 dBASE Ⅲ+의 Modify Command로 작성한 프로그램 화일에 한글을 입력했던 행은 동일 화일의 수정을 위하여 Modify Command를 재사용한 경우 화일이 깨어져 빈번히 다시 입력하곤 합니다(ⓐ 10, 30 '아무키나 누르시오'에서의 한글). 이유와 대책을 알고 싶습니다.

김태균/부산시 금정구 서3동 76

PE2 에디터를 사용하시면 됩니다.

dBASE Ⅲ+에서 삼보한글을 사용하는 경우 이와 같은 현상은 한글카드의 이상이 아니고 dbase Ⅲ+의 Command 인터프리터 이상입니다.

대체로 영문 소프트웨어의 일부는 MSB를 1로 세팅하는 아스키코드 값 128이상의 그래픽 문자에 대하여 배타적인 것들이 있는데 dBASE Ⅲ+ 자체 에디터도 그러한 종류중의 하나입니다. 따라서 그래픽 문자를 한글에 많이 이용하고 있는 삼보한글을 자체 에디팅 작업후 기억시킬 경우는 정상적인 한글의 표시 및 프로그램 운용이 어렵습니다. 이에 대한 대책은 다른 에디터를 사용하면 됩니다. 다음과 같은 내용을 담은 config.db라는 텍스트 화일을 작성한 후 dBASE를 실행시키고 Modify Command 명령을 주십시요. 이때 PE2 에디터는 패스로 묶여 있거나 같은 디렉토리에 있어야 합니다. 여기서 에디터가 굳이 PE2일 필요가 없으며 에디터 실행화일이름이면 됩니다.

```
TEDIT = PE2
```

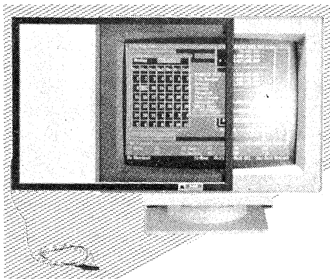
이러한 작업은 Modify Command 명령을 내렸을 때 도스상태에서 'PE2'라고 입력시킨 것과 똑같은 효과를 가집니다.

여기서 PE2 에디터를 권하는 이유는 PE2는 그래픽 캐릭터를 손상시키지 않으므로 한글이 깨지는 일이 없기 때문입니다.

또한 PE2는 매크로 에디터로 Alt키를 할당하기 어려운 삼보한글카드에서 에디트를 재정의해서 취향에 맞도록 설정할 수 있고, 블록명령이 잘되어 있어 라인정렬에 뛰어난 기능을 제공해 줍니다.

그러나 이렇게 작성된 프로그램 화일이라 하더라도 dBASE Ⅲ+의 명령어 인터프리터를 거쳐야 합니다. 이때 주의해야 하는 그래픽 문자가 있는데 그것은 아스키 138번으로 dBASE Ⅲ+의 명령어 인터프리터는 이 문자를 발견해도 없는 것처럼 해석합니다. 따라서 이 문자가 관여하는 '괴, 괸, 굉, 교, 구, 국, 군, 굳, 굴, 굵, 굶, 굼, 굽, 굿, 궁, 궈, 궐, 궤, 귀, 귓, 볽' 등의 한글을 표현하지 못합니다. 여기서 굳이 표현할 수 없는 문자를 표현하고 싶다면 한글 코드변환 및 디버그 등에서 문제점이 있기는 하지만 'CHR(138)+CHR(136) 으로 한글을 표현할 수 있읍니다. 한가지 기억할 것은 dBASE Ⅲ+의 I/O 문제는 프로그램화일에 국한되며 데이터 화일에는 아무런 영향을 주지 않습니다.

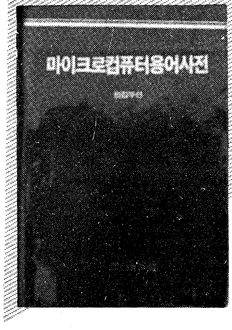
애독자 선물



3명

폴리에스터에 특수 구리도금과
아크릴 수지로 가공 처리된
첨단기술의 V.D.T 증후군
방지용 컴퓨터 보안기이다.

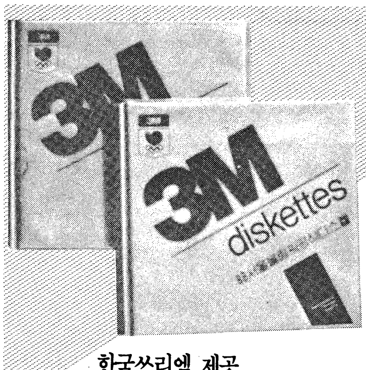
삼종실업 제공
TEL : 784-6931



5명

누구나가 한권씩은 지니고 있어야
하는 컴퓨터 용어사전으로 비교적
자세히 설명되어 있다.

오므사 제공
TEL : 754-1044



3명

특수 산화물 코딩 처리된
에러율 0%의 디스켓이다.

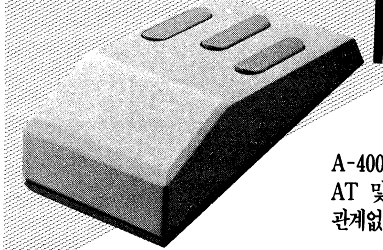
한국쓰리엠 제공
TEL : 785-4567



5명

컴퓨터에 전문지식이 없는 사람도
쉽게 접근할 수 있도록
꾸며진 좋은 지침서이다.

영진출판사 제공
TEL : 719-1119



1명

A-400 마우스는 IBM PC/XT/
AT 및 호환기종이면 기종에
관계없이 사용이 가능하다.

하나컴퓨터 제공
TEL : 717-4894

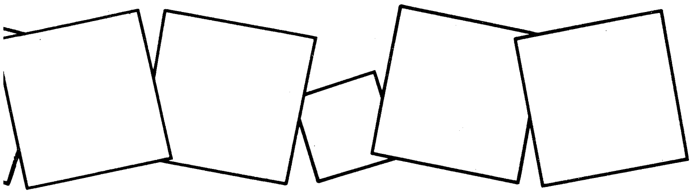


5명

미국 UL규격을 인정받은 제품으로
아크릴 시트로 특수가공되어
선명한 화면을 유지시켜준다.

화신전자 제공
TEL : 733-3567

대잔치





5명

SKC 제공
TEL : 756－5151

균일한 자성체의 배향과 성장으로 도막의 균일도 및 기록신호의 안정도를 극대화하였다.

월간 퍼스널컴퓨터는
독자 여러분의 의견을 소중히 하여
책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여
질의내용에 성의껏 답하시어
15일까지 보내주시면 27명을 뽑아
선물을 드립니다.

※ 서울 거주자는 5월31일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠읍니다.
기간은 5월31일까지이며, 이후는 무효로 처리합니다.

※ 애독자 설문지는 237페이지에 있읍니다.

애독자 선물대잔치
4월호 당첨자 명단

● 컴퓨터 보안기

김태형 서울 성북구 정릉2동 411-37
박기홍 제주도 남제주군 표선면 표선리 666
천상아 서울 구로구 구로동 317-11

● MS-DOS 입문

김교환 경남 울산시 무지동 855-20 (김익조씨댁)
김한수 서울 구로구 구로3동 구로APT 1동 403호
성정용 서울 관악구 신림9동 250-18
유응제 경기도 과천시 부림동 주공 APT 908동 403호
황명석 인천시 북구 작전동 한양APT 1동 702호

● 디스켓 박스(10장들이)

김진신 수원시 연무동 파송지구 68블럭 8노트 33통 1반
정호영 서울 노원구 공릉2동 348번지(신우연립 1동 102호)
조재완 경기도 안양시 안양2동 845-18 백광APT 302호
조성환 경북 안동시 송천동 국립안동대학교 물리학과 3학년
차수강 서울 양천구 목동APT 1331-1002

● 마우스

윤종호 서울 형당구 형당1동 122번지 11호 10/7

● 디스켓 자켓(4 장들이)

김동춘 서울 도봉구 창1동 731-69
박주욱 서울 영등포구 여의도동 한양APT D동 207호
엄윤성 강원도 원주시 단계동 777-5 10/1

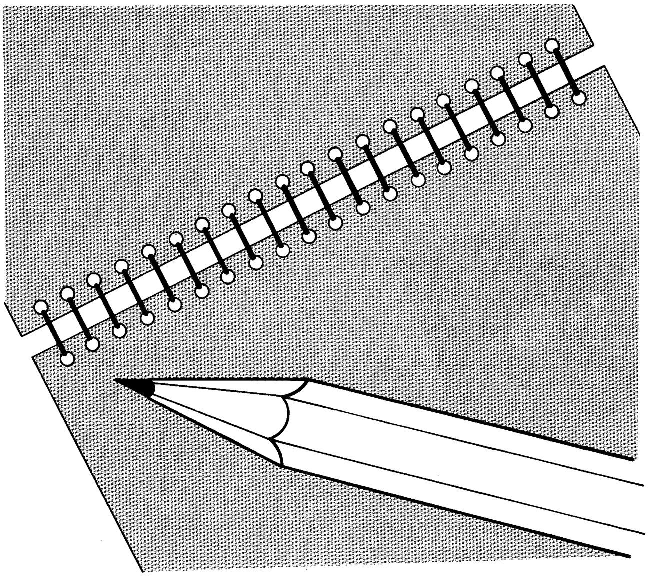
● 컴퓨터 용어사전

나은민 대전시 중구 문화동 문화맨션 가동 1104호
송상태 서울 성동구 행당동 17번지
이용주 부산시 남구 문현2동 530-8 7/5
이정열 서울 강서구 화곡본동 24-522
차원일 서울 성북구 석관2동 338-416 5/5

● 디스켓 보관함

권혁진 부산시 동구 수정동 1997 번지
김영국 서울 종로구 명륜동 324-100
박현주 서울 마포구 도화동 176-67
손기욱 서울 송파구 신천동 진주APT 10동 903호
장기태 경기도 수원시 성균관대학교 정보공학과 대학원

모뎀



컴퓨터 마인드 확산으로 정보와 데이터의 분배가 요구되고 있다. 갖고 있는 정보의 효율성을 극대화하기 위해서는 정보의 분배가 원활하게 이루어져야 하는데 정보의 분배에 획기적인 기여를 하는 것이 컴퓨터간의 통신이다.

컴퓨터간의 통신을 이용하여 주식정보, 데이타 통신의 처리안 등 여러가지의 정보를 제공받을 수 있고 이러한 서비스 업무를 제공받기 위해서는 반드시 모뎀을 사용해야 한다. 현재 모뎀은 전자기술의 발달로 과거의 값비싼 모뎀에서 얻을 수 없었던 신뢰성과 빠른 속도를 제공하고 있다.

모뎀의 전송속도

현재 모뎀의 성능을 평가할 수 있는 것은 전송속도라고 할 수 있다. 또한 모뎀은 전송속도에 따라 분류되어진다. 모뎀의 전송속도는 BPS(Bit Per Second)로 나타내는데 이는 초당 전송되는 비트의 수로 일반적으로 퍼스널컴퓨터에서는 1200bps의 전송속도를 갖는 모뎀이 많이 사용되고 있으나 앞으로는 2400 bps의 모뎀이 주류를 이룰 것으로 전망되고 있다. 이외의 4800, 9600bps의 전송속도를 갖는 고속 모뎀도 있다.

간단한 회보나 공공서비스 정도의 전송은 1200bps 모뎀 정도로도 충분하다. 그러나 중앙에서 각 지사로 광대한 데이터를 전송할 필요가 있는 경우는 고속 모뎀의 사용이 더 효율적이다.

내장형과 외장형 모뎀

퍼스널컴퓨터에 사용되는 모뎀은 설치하는 위치에 따라 외장형과 내장형으로 구분된다.

내장형 모뎀은 PC의 확장슬롯에 확장카드처럼 삽입하여 모뎀 자체의 공간을 차지하지 않고 사용되는데 비해 외장형 모뎀의 경우는 PC의 외부에 설치하여야 한다.

내장형 모뎀은 모뎀을 설치할 별도의 공간을 요구하지 않고 설치시에도 별도의 통신카드(RS-232C)를 필요로 하지 않는다. 그리고 별도의 전원을 요구하지 않고 PC자체 파워에서 공급되는 전원을 이용한다. 그러나 모뎀의 작동시 작동유무를 체크하기 곤란한 것이 단점이기도 하다. 그리고 대부분의 내장형 모뎀은 딥스위치를 이용하여 COM1, COM2 를 지정하여 사용할 수 있으나 COM1밖에 지정할 수 없는 내장형 모뎀은 다른 시리얼카드와 병행하여 사용할 수 없기 때문에 구입시 반드시 확인을 하는 것이 바람직하다.

또한 시리얼카드를 함께 사용할 수 있는 제품도 출하시에는 COM1으로 세팅되어 있기 때문에 다른 시리얼카드와 함께 사용할 경우에는 사용자 매뉴얼을 이용하여 COM2로 딥스위치를 조정시켜야 한다. 그러나 내장형 모뎀의 경우는 외장형 모뎀에 비하여 가격이 저렴하여 많이 이용되고 있고 앞으로 확산될 조짐을 보이고 있다.

외장형 모뎀의 경우는 내장형 모뎀보다 가격이 비싸기는 하나 모뎀의 동작상태를 수시로 파악할 수 있는 장점이 있다. 그리고 개인 용도의 서비스제공이 아닌 업무용으로는 외장형 모뎀을 사용하는 것이 바람직하다.

모뎀의 설치

모뎀의 설치방법을 간단히 설명하면 다음과 같다.

1. 내장형모뎀

1) 접속하고자 하는 컴퓨터의 케이스를 연다.
2) IBM-PC XT/AT의 슬롯중 빈 슬롯을 찾아 모뎀를 삽입한다.
3) 모뎀을 끼워 넣고난 후 브라켓을 PC면에 고정시킨다.
4) 컴퓨터의 케이스를 덮고난 후 제공된 모듈러 케이블을 모듈러 잭에 연결한다.
5) 전화기를 TEL 모듈러 잭에 연결한다.
6) 파워를 넣고 FDD에 통신용

프로그램을 넣고 부팅시킨다.
7) 제공된 모뎀 사용설명서의 명령어 안내서를 따라 모뎀을 테스트한다.

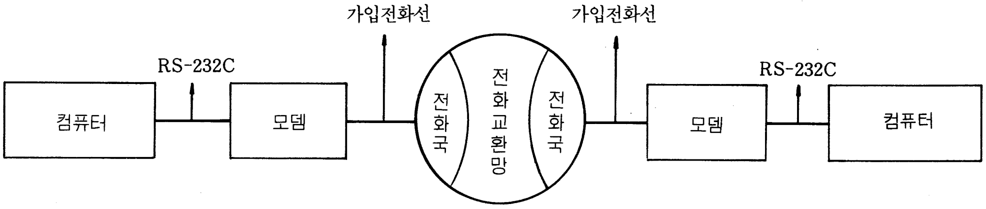
2. 외장형모뎀

1) 접속하고자 하는 컴퓨터의 케이스를 연다.
2) IBM-PC XT / AT의 슬롯중 빈 슬롯을 찾아 통신카드(RS-232C)를 삽입한다.
3) 통신카드(RS-232C)를 끼워 넣은 후 브라켓을 PC면에 고정시킨다.
4) 컴퓨터의 케이스를 덮은 후 제공된 모뎀케이블을 통신카드에 접속시킨다.
5) 전화기라인을 모뎀에 연결한다.
6) 파워를 넣고 FDD에 통신용 프로그램을 넣고 부팅시킨다.
7) 제공된 모뎀 사용설명서의 명령어 안내서를 따라 모뎀을 테스트한다.

모뎀의 규격

컴퓨터 통신을 하려는 사람이나 데이터 통신에 관심이 있는 사람이면 귀에 익은 내용이 BELL타입과 CCITT타입이다. 이것은 모뎀의 동작규격을 분류하는 것으로 BELL타입은 미국의 전화회사인 AT&T(American Telegraph & Telephone)에서 정한 통신 규격이고 이 규격은 미국을 중심으로 사용되고 있다. CCITT타입은 국제전신전화 자문위원회(Consultative Committee of International Telegraph & Telephone) 에서 정한 규격으로 세계적으로 사용되고 있으며, 유럽을 중심으로 한 많은 국가에서 이 규격을 따르고 있고 우리나라도 이 방식을 선택하고 있다. 따라서 모뎀을 구입시에는 두가지 규격을 만족하는 모뎀을 선택하는 것이 바람직한 방법일 것이다.(대부분의 모뎀의 경우는 BELL과 CCITT타입을 선택하는 스위치를 갖고 있다) 모뎀은 상대의 규격이 다르면 통신이 불가능하므로 규격을 충분히 검토하여야 한다.

〔그림 1〕



체신부의 형식승인

모뎀의 형식승인은 체신부에서 시험을 통해 결정되고, 형식승인 내용에 적합한 사용을 하기 위해서는 형식승인을 받은 업체의 모뎀을 사용하는 것이 바람직하다. 현재 대부분의 내장형 모뎀은 형식승인을 받지 못하고 있는데 최근에 형식승인을 받은 제품이 공급되고 있어 내장형 모뎀의 수요창출을 유도할 것으로 전망된다.

모뎀의 종류

모뎀의 종류는 기능에 따라 3가지로 분류될 수 있는데 각각의 특징을 간략하게 기술하면 다음과 같다.

더미(Dummy)모뎀은 모뎀의 기본기능에다 약간의 제어기능이 추가된 것으로 설치가 간단하고 가격이 저렴하나 고품질의 통신 및 부대기능 수행에는 부적합하여 현재는 전용선 일부에서만 사용하고 있다.

스마트(Smart)모뎀은 인터페이스로부터 자동호출, 회선선택, 공중데이터 통신이나 DMA기능을 내장하여 신뢰성있는 데이터통신을 할 수 있어 자동화를 실현할 수 있으나 가격이 비싼 것이 흠이다.

인텔리전트(Intelligent) 모뎀은 자체에 마이크로 프로세서를 내장하여 상황 판단을 능동적으로 하며, 통신속도변환, 프로토콜 변환 등이 자동적으로 처리되는 기능을 보유하고 있다. 아직은 고가인 관계로 널리 보급되지 못하고 있으나 편리성, 정확성, 높은 신뢰성을 갖는 통신기기로서의 강점때문에 향후 모뎀시장의 주류를 이룰 것이다.

모뎀의 운용방식

통신회선을 이용한 정보 유통로 구성은 크게 구별하여 전화교환회선과 전용회선으로 구분된다. 전화 교환회선은 **그림** 1과 같이 전화형 모뎀을 통해 가입 전화선에 연결된다. 이 방식은 대체로 요구되는 통신시간이 적을때 많이 사용되고 있다.

전용회선은 통신시간설정의 부담없이 하루 24시간 항상 정보 통신이 가능하며 대체로 요구되는 정보 통신시간이 많을때 이용된다.

1200BPS이하

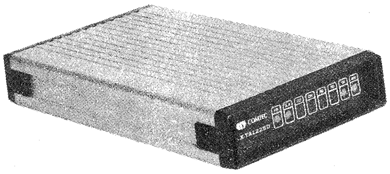
MICROMAX 1200PC

- 공급처 : 하이맥스
- 전화 : (02)540–6253
- 전송속도 : 1200bps
- 전송방식 : 비동기식
- 회선형식 : 전이중 / 반이중 방식
- 권고채택 : CCITT V.22 / Bell 212A
- 가격 : 15만원(VAT 포함)

MICROMAX 2400PC

- 공급처 : 하이맥스
- 전화 : (02)540–6253
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 회선형식 : 전이중 / 반이중
- 권고채택 : CCITT V.22 / V.21, Bell 103 / 212A
- 가격 : 19만원(VAT 포함)

GTM-1212PC2
- 공급처 : 자네트 시스템
- 전화 : (02)546-4741
- 전송속도 : 1200bps
- 전송방식 : 동기식 / 비동기식
- 회선형식 : 공중회선용 2선식 전이중 방식
- 권고채택 : Bell 103 / 212A, CCITT V.21 / V.22

X-TAL22BD


- 공급처 : 콤텍시스템
- 전화 : (02)757-0520
- 전송속도 : 1200 / 300bps
- 전송방식 : 비동기식
- 변조방식 : 주파수 편이방식
- 회선형식 : 전이중 방식 / 반이중 방식
- 권고채택 : CCITT V.22 / V.21 Bell 212A / 103

DC-1200


- 공급처 : 데이타 콤
- 전화 : (02)242-2191
- 전송속도 : 1200bps
- 전송방식 : 비동기식
- 변조방식 : 주파수 편이방식
- 회선형식 : 2선식 및 4선식 전용선
- 권고채택 : CCITT V.23

GTM-1212VI
- 공급처 : 자네트시스템
- 전화 : (02)546-4741
- 전송속도 : 1200 / 600bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 차등위상 편이방식
- 회선형식 : 2선식 전이중방식
- 권고채택 : CCITT V.22, Bell 212A / 103

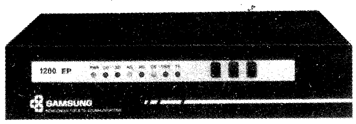
GTM-1200VI
- 공급처 : 자네트시스템
- 전화 : (02)546-4741
- 전송속도 : 1200bps
- 전송방식 : 비동기
- 변조방식 : 주파수 편이방식
- 회선형식 : 2선식 반이중방식, 4선식 전이중방식
- 권고채택 : CCITT V.23

SST-1200AD
- 공급처 : 삼성전자
- 전화 : (02)559-3114
- 전송속도 : 1200 / 300bps
- 전송방식 : 비동기식
- 변조방식 : 위상편이 방식
- 회선형식 : 2선식 전이중방식
- 권고채택 : CCITT V.22, Bell 103 / 212A
- 가격 : 50만원(VAT 별도)

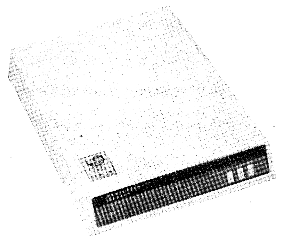
SST-1200DU
- 공급처 : 삼성전자
- 전화 : (02)559-3114
- 전송속도 : :1200 / 600bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 위상편이방식
- 회선형식 : 2선식 이중방식
- 권고채택 : CCITT V.22
- 가격 : 30만원(VAT 별도)

SST-301AD
- 공급처 : 삼성전자
- 전화 : (2)559-3114
- 전송속도 : 300bps
- 전송방식 : 비동기식
- 변조방식 : 주파수 편이방식
- 회선형식 : 2선식 전이중방식
- 권고채택 : CCITT V.21
- 가격 : 25만원(VAT 별도)

SST-300DU
- 공급처 : 삼성전자
- 전화 : (02)559-3114
- 전송속도 : 300bps
- 변조방식 : 주파수 편이방식
- 회선형식 : 2선식 전이중방식
- 권고채택 : CCITT V.21
- 가격 : 15만원(VAT 별도)

SST-1200EP


- 공급처 : 삼성전자
- 전화 : (02)559-3114
- 전송속도 : 600 / 1200bps
- 전송방식 : 비동기식
- 변조방식 : 주파수 편이방식(FSK)
- 권고채택 : CCITT V.23
- 가격 : 15만원(VAT 별도)

GSM-12N


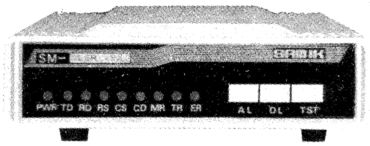
- 공급처 : 금성반도체(주)
- 전화 : (02)787-2923
- 전송속도 : 1200bps
- 전송방식 : 비동기식
- 변조방식 : 주파수 편이(FSK)방식
- 회선형식 : 2선식 및 4선식 전용회선
- 권고채택 : CCITT V.23
- 가격 : 12만원(VAT 별도)

GSM-1212
- 공급처 : 금성반도체
- 전화 : (02)787-2923
- 전송속도 : 1200bps
- 변조방식 : DPSK
- 회선형식 : 2선식 FDX / HDX 교환



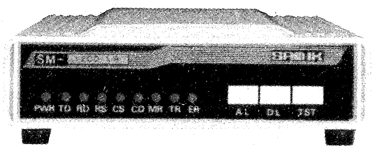
회선
- 권고채택 : CCITT V.22, Bell 103 / 212A
- 가격 : 27만원(VAT 별도)

SM-LSR192



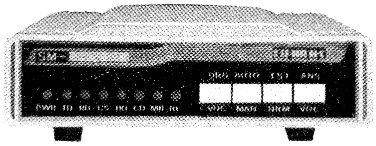
- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 1200 / 600bps
- 전송방식 : 동기식
- 변조방식 : 복극성
- 회선형식 : 반이중방식 / 전이중방식
- 가격 : 13만원(VAT 별도)

SM-1200LA



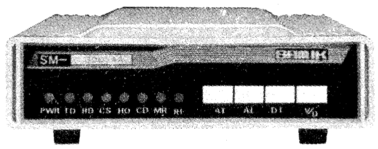
- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 1200 / 600bps
- 전송방식 : 비동기
- 변조방식 : 주파수 편이
- 권고채택 : CCITT V.23
- 가격 : 13만원(VAT 별도)

SM-1200AD



- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 위상변조방식
- 권고채택 : CCITT V.22B
- 가격 : 30만원(VAT 별도)

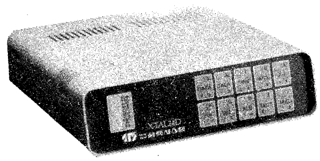
SM-300AA



- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 300bps
- 전송방식 : 비동기방식
- 변조방식 : 주파수 변이
- 권고채택 : CCITT V.21
- 가격 : 20만원(VAT 별도)

2400BPS이하

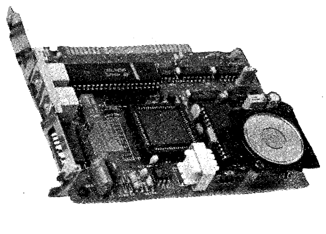
X-TAL24D



- 공급처 : 콤텍시스템
- 전화 : (02)715-4911
- 전송속도 : 1200 / 2400bps
- 전송방식 : 동기식 / 비동기식
- 회선형식 : 전이중방식
- 권고채택 : Bell 103 / 113, CCITT V.22A/V.22B

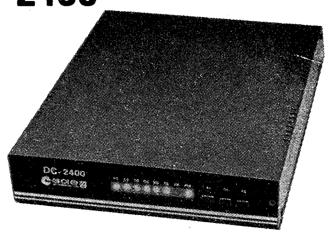
X-TAL24LDI

- 공급처 : 콤텍시스템
- 전화 : (02)715-4911
- 전송속도 : 2400/1200bps/300bps
- 전송방식 : 비동기식



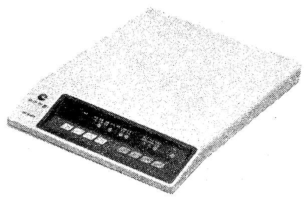
- 회선형식 : 전이중방식
- 권고채택 : CCITT V.22, Bell 212A / 103

DC-2400



- 공급처 : 데이타 콤
- 전화 : (02)242-2191
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식
- 변조방식 : 위상편이방식
- 회선형식 : 4선식 전용선
- 권고채택 : CCITT V.26

DC-2422



- 공급처 : 데이타 콤
- 전화 : (02)242-2191
- 전송속도 : 2400 / 1200bps
- 전송속도 : 동기식 / 비동기식
- 회선형식 : 전이중 2선식 교환회선
- 권고채택 : CCITT V.22 / V.21, Bell 212A / 103

GTM-2424VI

- 공급처 : 자네트시스템
- 전화 : (02)546-4741
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식



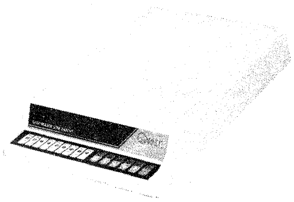
- 변조방식 : 차등위상 편이방식
- 회선형식 : 2선식 전이중방식
- 권고채택 : CCITT V.22, Bell 212A / 103

GTM-2424PI



- 공급처 : 자네트시스템
- 전화 : (02)546-4741
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 차등위상편이(DPSK)
- 회선형식 : 동기식 / 비동기식
- 권고채택 : CCITT V.21 / V.22, Bell 212A / 103

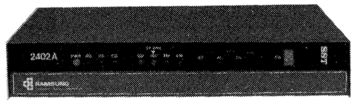
GTM-2400VI



- 공급처 : 자네트시스템
- 전화 : (02)546-4741
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 차등위상편이 방식
- 회선형식 : 반이중 / 전이중(4선식) 방식
- 권고채택 : CCITT V.26 / V.54, Bell 201

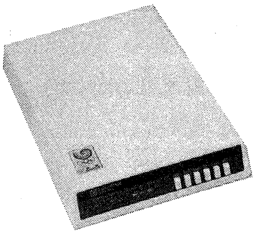
SST-2402A

- 공급처 : 삼성전자



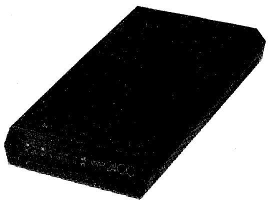
- 전화 : (02)559-3114
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식
- 변조방식 : 위상 편이방식
- 권고채택 : CCITT V.26
- 가격 : 25만원(VAT 별도)

GSM-24N



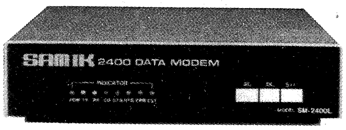
- 공급처 : 금성반도체(주)
- 전화 : (02)787-2923
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 차등위상 편이방식
- 회선형식 : 2선식 및 4선식 전용회선
- 권고채택 : CCITT V.26, Bell-201 B / C
- 가격 : 25만원(VAT 별도)

GSM2400



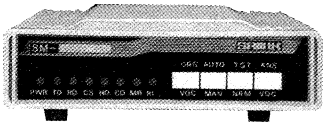
- 공급처 : 금성반도체(주)
- 전화 : (02)787-2923
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : QAM
- 회선형식 : 2선식 FDX / HDX 교환회선
- 권고채택 : CCITT V.22, Bell 212A
- 가격 : 37만원(VAT 별도)

SM-2400L



- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식
- 변조방식 : 위상변조방식
- 권고채택 : CCITT V.26
- 가격 : 28만원(VAT 별도)

SM-2400AD



- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 2400 / 1200bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 위상변조방식
- 권고채택 : CCITT V.22, Bell 212A
- 가격 : 45만원(VAT 별도)

4800BPS이하

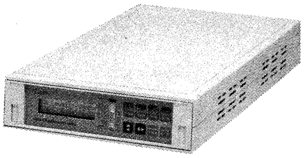
DC-48LSI



- 공급처 : 데이타 콤
- 전화 : (02)242-2191
- 전송속도 : 4800bps
- 전송방식 : 동기식
- 변조방식 : 8상위상 편이방식
- 회선형식 : 2선식 / 4선식 전용선
- 권고채택 : CCITT V.27

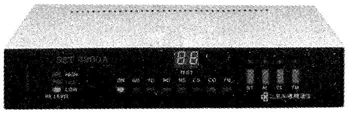
GTM-4801

- 공급처 : 자네트시스템



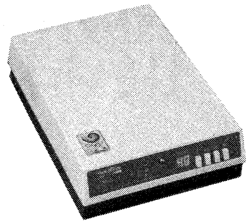
• 전화 : (02)546-4741
• 전송속도 : 4800 / 2400bps
• 전송방식 : 동기식
• 변조방식 : 차등위상 편이방식
• 변조방식 : 4선식전이중, 2선식 반이중방식
• 권고채택 : CCITT V.27

SST 4800A



• 공급처 : 삼성전자
• 전화 : (02)559-3114
• 전송속도 : 4800 / 2400bps
• 전송방식 : 동기식
• 변조방식 : 위상편이방식
• 권고채택 : CCITT V.27
• 가격 : 130만원(VAT 별도)

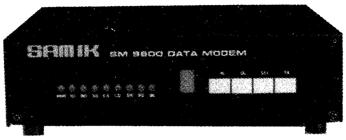
GSM-48N



• 공급처 : 금성반도체(주)
• 전화 : (02)787-2923
• 전송속도 : 4800 / 2400bps
• 전송방식 : 동기식 / 비동기식
• 변조방식 : 차등위상 편이(DPSK)방식
• 권고채택 : CCITT V.27
• 회선형식 : 2선식 및 4선식 전용회선
• 가격 : 130만원(VAT 별도)

SM-MPS4827

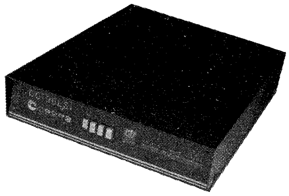
• 공급처 : 삼익전자공업(주)



• 전화 : (02)294-7171
• 전송속도 : 4800 / 2400bps
• 전송방식 : 동기식
• 변조방식 : 위상변조방식
• 권고채택 : CCITT V.27
• 가격 : 110만원(VAT 별도)

4800BPS이상

DC-96LSI



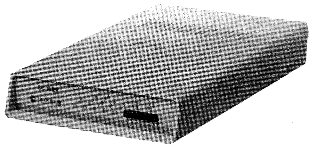
• 공급처 : 데이타 콤
• 전화 : (02)242-2191
• 전송속도 : 9600bps
• 전송방식 : 동기식
• 변조방식 : 8상 4진폭 변조방식
• 회선형식 : 2선식 및 4선식 전용선
• 권고채택 : CCITT V.29

DC-200LD

• 공급처 : 데이타 콤
• 전화 : (02) 242-2191
• 전송속도 : 900/1200/1800/2400/3600/4800/7200/9600bps
• 전송방식 : 비동기식 / 동기식
• 회선형식 : 2선식 반이중방식 / 4선식 전이중방식
• 권고채택 : CCITT V.24

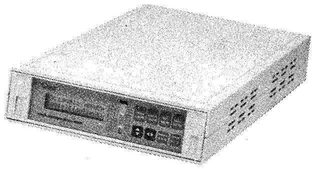
DC-1922

• 공급처 : 데이타 콤
• 전화 : (02)242-2191
• 전송속도 : 1200 / 2400 / 4800 / 9600 / 19200bps



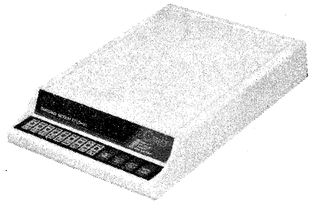
• 전송방식 : 동기식 / 비동기식
• 회선형식 : 4선식 전이중방식 / 2선식 반이중방식
• 권고채택 : CCITT V.24

GTM-9601



• 공급처 : 자네트시스템
• 전화 : (02)546-4741
• 전송속도 : 9600 / 7200 / 4800 / 2400bps
• 전송방식 : 동기식
• 변조방식 : QAM / DPSK
• 회선형식 : 4선식 전이중, 2선식 반이중방식
• 권고채택 : CCITT V.29 / V.27

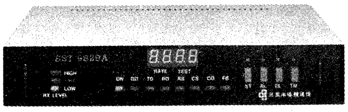
GTLD-VI



• 공급처 : 자네트시스템
• 전화 : (02)546-4741
• 전송속도 : 1200 / 2400 / 4800 / 7200 / 9600 / 14400 / 19200bps
• 전송방식 : 동기식 / 비동기식
• 변조방식 : 위상변조방식
• 회선형식 : 4선식 전이중, 2선식 반이중방식
• 권고채택 : CCITT V.24 / V.28

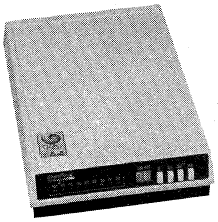
SST 9629A

• 공급처 : 삼성전자



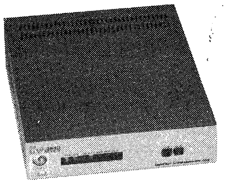
- 전화 : (02)559-3114
- 전송속도 : 9600 / 7200 / 4800bps
- 전송방식 : 동기식
- 변조방식 : 진폭변조방식
- 회선형식 : 4선식 전용선
- 권고채택 : CCITT V.29
- 가격 : 150만원(VAT 별도)

GSM-96N



- 공급처 : 금성반도체(주)
- 전화 : (02)787-2923
- 전송속도 : 9600 / 7200 / 4800bps
- 전송방식 : 동기식 / 비동기식
- 변조방식 : 진폭변조(QAM)방식
- 회선형식 : 2선식 및 4선식 전용회선
- 권고채택 : CCITT V.29
- 가격 : 150만원(VAT 별도)

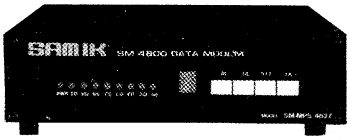
LD-3840



- 공급처 : 금성반도체
- 전화 : (02)787-2923
- 전송속도 : 2400 / 4800 / 9600 / 19200bps
- 전송방식 : 동기식 / 비동기식
- 회선형식 : 4선식 전용회선
- 최대사용거리 : 1km
- 가격 : 120만원(VAT 별도)

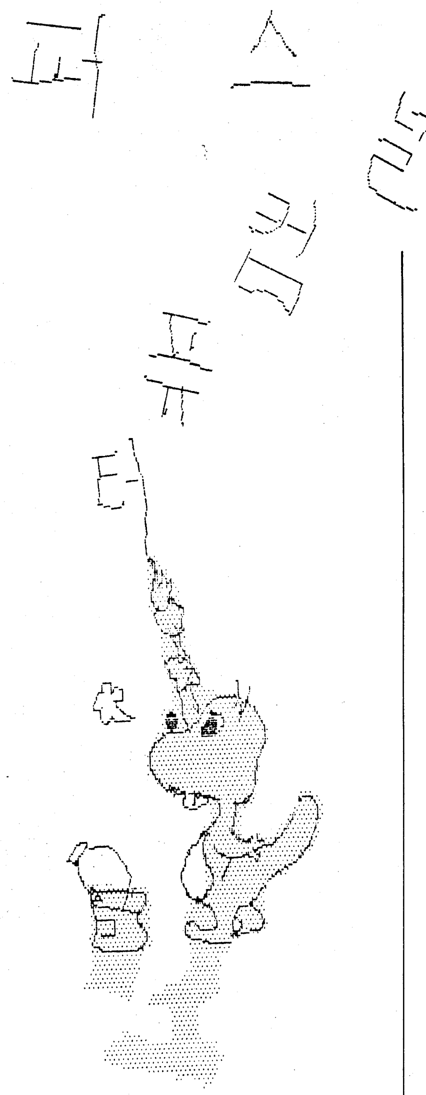
- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 600 / 19200bps
- 전송방식 : 비동기식
- 변조방식 : 복극성
- 회선형식 : 반이중방식 / 전이중방식

SM-9600



- 공급처 : 삼익전자공업(주)
- 전화 : (02)294-7171
- 전송속도 : 9600 / 7200 / 4800bps
- 전송방식 : 동기식
- 변조방식 : 위상 진폭변조방식
- 권고채택 : CCITT V.29
- 가격 : 120만원(VAT 별도)

책을 마감하고



• 최근 데이타통신의 네트워크에 바이러스 경고가 발생하자 데이타통신에서는 비상상태였었다. "이젠 정말 정보화시대로 가나보다"라는 낙관론을 여지없이 무너뜨리고 정보화시대의 후면을 공격한 이 바이러스의 등장은 이미 오래된 일이었다. 그러나 그저 조심할 수 밖에 없는 것이 현재로서는 최선의 방어일뿐이다.

정보화 사회니, 시대의 도래니하며 모든 것을 편하고자 하는 것들은 환영할 일이지만 자신이 최고라는 우월성을 과시한다든지 남들이 미숙하게 사용한다고 해서 파괴하려는 사회가 된다는 것은 아예 진보가 필요없는 것이다. 사회가 진보해 가고 있는 만큼 도덕성도 진보할 수 있다면 테크노피아의 이상은 정말 멀지 않을텐데. 〈둘리〉

• 여섯!

내일모레면 '전자과학'의 30주년 기념식을 갖는다.

30여년 동안 국내 전자산업의 지식시장을 묵묵히 형성해 온 거목과 이제 새싹을 부지런히 움트며 돋아나는 '퍼스널컴퓨터'가 좋은 앙상블을 이룬다.

핑계에 30년후, 아니 그 이상의 '퍼스널컴퓨터'를 위해 오늘은 어떻게 보낼까? 하고 생각해 보건데 역시 '항상 독자와 함께 공부하는 자세'와 '항상 겸손한 자세'가 내가 지녀야 할 바람직한 몸가짐인 것 같다.

30년의 영광을 안은 전자과학의 선배들이 자랑스럽게 보이며 난 오늘도 여지없이 게기고(?) 싶다. 밤을 단잠으로 보내기엔 독자들의 시선이 너무 매섭기도 하고… 〈榮〉

• 새로운 벗을 만났으니 기쁘고 또 기뻐해야겠는데, 그 뒤에 또 다른 헤어짐이 기다리고 있다고 생각하니 오히려 이 싱그러운 봄내음마저 싸늘하게 느껴집니다.

이럴 때는 회사 옥상에라도 올라가 정면으로 남산을 바라보며 크게 외쳐볼까. "남산아 따뜻한 나의 품안으로… 올테면 오라지…" 〈김현〉

• 5월은 계절의 여왕이라 하는데
우리는 5월을 계절의 마왕이라 한다.
무척이나 맵게 보낸 지난 5월
5월이면 슬퍼지는 사람들
5월에 잔인했던 사람들
5월에 인생을 종료한 사람들
암울했던 지난 5월
5월에 불행했던 사람들
이 모두에게 고개숙여 애도하면서
이 초라한 글을 스쳐지나간
진아에게 처음이자 마지막으로 받친다.
〈石心〉

산다는 명제앞에
가슴뛰는 절박함도 만나지겠지만,
어디곳 어디서든
나의 진실한
표상이고 싶다. 〈淑〉

• 막연히 매년마다 맞이하는 오월이지만
언제부턴가
찬란하기 보다는
가슴을 휘젖고 지나는 계절로
와닿는다.
올해에도 다가온 오월은
교과서에서나 읊어대던
신록예찬이기 전에,
말로만 듣던 보리고개보다
더 험한 모습으로 다가온다.
나, 혼자만 느낀 감상이라면
오죽이나 좋을텐데
이게 웬일인가?
당장, 먹고 살 양식은 있다해도
배불리 먹어도 먹어도
차지않는 이 갈증을… 〈鎔〉